VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Andrej BUGAJEV

THE INVESTIGATION OF EFFICIENCY OF PHYSICAL PHENOMENA MODELLING USING DIFFERENTIAL EQUATIONS ON DISTRIBUTED SYSTEMS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (07T)

VGTU
LEIDYKLA
TECHNIKA
Vilnius 2015

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2011–2015.

Supervisor

Prof. Dr Habil. Raimondas ČIEGIS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

The Dissertation Defense Council of Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Habil. Antanas ČENYS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

Members:

Prof. Dr Habil. Rimantas BARAUSKAS (Kaunas University of Technology, Informatics – 09P),
Prof. Dr Habil. Gintautas DZEMYDA (Vilnius University, Informatics Engineering – 07T),
Prof. Dr Arvet PEDAS (University of Tartu, Mathematics – 01P),
Prof. Dr Olegas VASILECAS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of Informatics Engineering in the Senate Hall of Vilnius Gediminas Technical University at **9 a. m. on 27 November 2015**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.
Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vgtu.lt

A notification on the intend defending of the dissertation was send on 26 October 2015.

A copy of the doctoral dissertation is available for review at the VGTU repository http://dspace.vgtu.lt and at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania).

VGTU leidyklos TECHNIKA 2341-M mokslo literatūros knyga

ISBN 978-609-457-857-1

zvex77777@gmail.com

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Andrej BUGAJEV

FIZIKINIŲ REIŠKINIŲ MODELIAVIMO NAUDOJANT DIFERENCIALINES LYGTIS EFEKTYVUMO PASKIRSTYTOSE SISTEMOSE TYRIMAS

MOKSLO DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2015

Disertacija rengta 2011–2015 metais Vilniaus Gedimino technikos universitete.

Vadovas

prof. habil. dr. Raimondas ČIEGIS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. habil. dr. Antanas ČENYS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Nariai:

prof. habil. dr. Rimantas BARAUSKAS (Kauno technologijos universitetas, informatika – 09P),
prof. habil. dr. Gintautas DZEMYDA (Vilniaus universitetas, informatikos inžinerija – 07T),
prof. dr. Arvet PEDAS (Tartu universitetas, matematika – 01P),
prof. dr. Olegas VASILECAS (Vilnius Gedimino technikos universitetas, informatikos inžinerija – 07T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje 2015 m. lapkričio 27 d. 9 val. Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.
Tel.: (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vgtu.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2015 m. spalio 26 d.

Disertaciją galima peržiūrėti VGTU talpykloje http://dspace.vgtu.lt ir Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva).

Abstract

This work is dedicated to development of mathematical modelling software. In this dissertation numerical methods and algorithms are investigated in software making context. While applying a numerical method it is important to take into account the limited computer resources, the architecture of these resources and how do methods affect software robustness. Three main aspects of this investigation are that software implementation must be efficient, robust and be able to utilize specific hardware resources. The hardware specificity in this work is related to distributed computations of different types: single CPU with multiple cores, multiple CPUs with multiple cores and highly parallel multithreaded GPU device.

The investigation is done in three directions: GPU usage for 3D FDTD calculations, FVM method usage to implement efficient calculations of a very specific heat transferring problem, and development of special techniques for software for specific bacteria self organization problem when the results are sensitive to numerical methods, initial data and even computer round-off errors. All these directions are dedicated to create correct technological components that make a software implementation robust and efficient.

The time prediction model for 3D FDTD calculations is proposed, which lets to evaluate the efficiency of different GPUs. A reasonable speedup with GPU comparing to CPU is obtained. For FVM implementation the OpenFOAM open source software is selected as a basis for implementation of calculations and a few algorithms and their modifications to solve efficiency issues are proposed. The FVM parallel solver is implemented and analyzed, it is adapted to heterogeneous cluster Vilkas. To create robust software for simulation of bacteria self organization mathematically robust methods are applied and results are analyzed, the algorithm is modified for parallel computations.

Reziumė

Šitas darbas skirtas matematinio modeliavimo programinei įrangai kurti. Disertacijoje ištirti skaitiniai metodai ir algoritmai programinės įrangos kūrimo kontekste. Kai taikomas skaitinis metodas yra svarbu atsižvelgti į ribotus kompiuterių išteklius, šių išteklių architektūrą ir metodų įtaką programinės įrangos patikimumui. Trys pagrindiniai tyrimų aspektai yra programinės įrangos efektyvumas, patikimumas ir galimybė panaudoti specialiuosius kompiuterinius išteklius. Darbe naudojami kompiuteriniai ištekliai, kuriems yra būtini skirtingų tipų paskirstytieji skaičiavimai: vienas procesorius su keliais branduoliais, daug procesorių turinčių po kelis branduolius ir stipriai lygiagretus daugiagijinis skaičiavimų įrenginys (GPU).

Tyrimas atliktas trimis kryptimis: GPU panaudojimas 3D BSLS skaičiavimams, BTM metodo panaudojimas efektyviai skaičiavimų realizacijai tam tikrais ypatumais pasižyminčiam šilumos laidumo uždaviniui spręsti ir nestabiliems modeliams skirtų sprendimų kūrimas – modeliuojamas specifinis bakterijų saviorganizacijos procesas, kai skaičiavimų rezultatai jautrūs skaitiniams metodams, pradinėms sąlygoms ir net apvalinimo paklaidoms. Visos šios kryptys yra skirtos taisyklingų technologinių komponentų, darančių realizaciją patikima ir efektyvia, kūrimui.

Darbe pasiūlytas modelis 3D BSLS skaičiavimų laikams prognozuoti, tai leidžia įvertinti tame tarpe ir pačių skaičiavimo įrenginių efektyvumą šitokios rūšies skaičiavimų atžvilgiu. Algoritmo pritaikymas ištestuotiems 3D BSLS skaičiavimų atvejams duoda apie 10 kartų pagreitėjimą lyginant su CPU lygiagrečiąja versija. BTM skaičiavimų realizacijos pagrindui pasirinktas atviro kodo paketas OpenFOAM, pasiūlyti keli metodai ir algoritmai efektyvumo problemoms spręsti. Realizuotas lygiagretusis BTM sprendiklis adaptuotas prie heterogeninio klasterio Vilkas. Patikimai programinei įrangai, skirtai bakterijų saviorganizacijos simuliacijai, kurti pasiūlyti ir pritaikyti matematiškai patikimi metodai. Atlikta skaičiavimų eksperimentų analizė, pateikta algoritmo lygiagrečioji versija.

Contents

INTRODUCTION 1
Motivation 1
Problem Formulation 4
Relevance of the Thesis 4
Research Object 5
The Aim of the Thesis 5
The Objectives of the Thesis 5
Research Methodology 6
Scientific Novelty of the Thesis 6
Practical Value of Research Findings 8
Defended Statements 9
Approval of Research Findings 9
Structure of the Dissertation 9
Aknowledgement 10

1. 3D FINITE-DIFFERENCE TIME-DOMAIN CALCULATIONS USING GRAPHICS PROCESSOR UNIT 11
1.1. Introduction Into First Chapter 12
1.2. Occupancy 16
1.3. The Template of Numerical Algorithms 18
1.4. Complexity Analysis 22

1.5. Application to a Real Problem – Propagation of TE_{10} Wave in Rectangular Waveguide 27
1.6. Summary of the First Chapter 31
1.7. Conclusions of the First Chapter 32

2. THE MODELLING OF HEAT TRANSFER IN ELECTRICAL CABLES 33
2.1. Introduction Into Second Chapter 34
2.2. Mathematical Modelling Problem 37
2.3. Modelling Approach 41
2.3.1. Finite Volume Method Software 41
2.3.2. The Benchmark to Control the Efficiency 42
2.3.3. Non-orthogonality Error Problem Solving 44
2.3.4. Adaptive Meshes 48
2.4. Introduction Into Parallel Computing Using OpenFOAM 65
2.5. Benchmark Problem 67
2.6. Parallel OpenFOAM-based Solver 68
2.7. Scalability Analysis of the Parallel Algorithm 69
2.8. Parallel Performance Tests and Analysis 71
2.9. Conclusions of the Second Chapter 76

3. THE METHODS FOR NON-STABLE PHYSICAL PROCESS MODELS 77
3.1. Introduction Into Third Chapter 78
3.2. The Finite Difference Scheme 86
3.3. Numerical Experiments 88
3.3.1. Convergence Analysis 88
3.3.2. Formation of Complex Patterns 89
3.4. Parallel Numerical Algorithms 91
3.5. Parallel Algorithm 94
3.5.1. Parallel Factorization Algorithm 96
3.5.2. Scalability Analysis 100
3.6. Computational Experiments 101
3.7. Summary of the Third Chapter 102
3.8. Conclusions of the Third Chapter 103

GENERAL CONCLUSIONS 105

REFERENCES 107

LIST OF SCIENTIFIC PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF THE DISSERTATION 115

SUMMARY IN LITHUANIAN 117

APPENDIXES[1] 129
Appendix A. Adaptive Meshes 130
Appendix A.1. The Galerkin finite element method 130
Appendix A.2. Mesh generation using duality-based technique . 131
Appendix A.3. Comparison with apriori adaptive meshes 133
Appendix B. The Finite Difference Scheme 140
Appendix B.1. The method of lines. Discretization in space . . . 140
Appendix B.2. Operator splitting methods 142
Appendix B.3. Numerical integration of ODEs 144
Appendix C. The Co-authors Agreements to Present Publications for the Dissertation Defence 147
Appendix D. The Copies of Scientific Publications by the Author on the topic of the Dissertation 148

[1]The annexes are supplied in the enclosed compact disc

Introduction

Motivation

Information technology progresses rapidly and the usage of its potential becomes more and more challenging. Computers become faster, new algorithms appear, software libraries become more sophisticated. One of the most important usage of these technologies deal with physical processes modelling in order to conduct physical experiments virtually. The general virtual experiment technology scheme is shown in Fig. 1. In this dissertation the main accent is done on three components of presented scheme:

- Numerical methods.
- Numerical model and algorithms.
- Software implementation.

This dissertation is devoted to improving virtual experiment technology which lets to perform a physical experiment virtually, using computer simulation of physical processes. We consider applications of modelling methods in a software making context. The main scientific problem is to successfully apply some recent technologies to new modelling problems. The following aspects are taken into account:

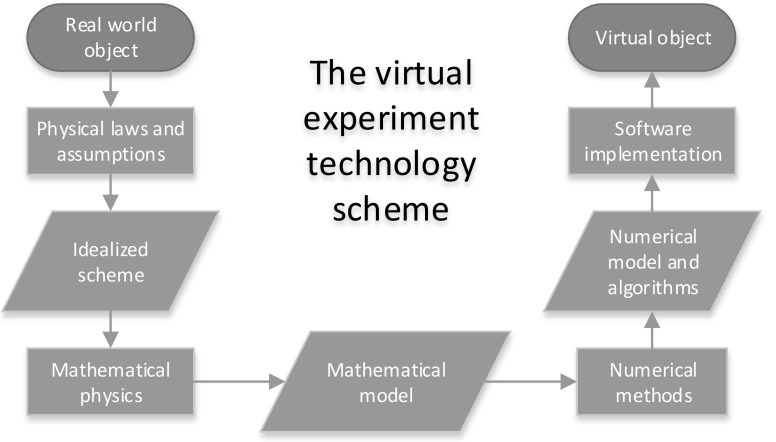


Fig. 1. The virtual experiment technology scheme

- How well the computer resources are utilized (converting the theoretical hardware possibilities into real computations).
- How efficient the application of a method is. It must satisfy the computational error criteria and be as fast as needed.
- How stable the obtained results are – they must be correct for different variations of a physical problem (e.g. material properties and geometry can vary).

It should be noted that there is no universal formula to answer all these questions – it strongly depends on specificity of a given problem. There are different physical processes with some unique specificities and different aspects of methods are important in different cases.

There exist many general modelling packages, based on Finite elements method (FEM), Finite volumes method (FVM) and Finite differences method (FDM). However, universal approaches can never guarantee that the problem will always be solved efficiently. The stability of results also depends on a user – he can select some options that often are sufficient to successfully solve some standard problems. However, for complicated problems it is typical to have specificity that makes problem hard to solve using universal packages. So there is nearly limitless space for improving the existing general-purpose packages and that is what this work is devoted to.

The implementation of physical process modelling can be achieved by performing several steps, each of them must be performed correctly in order to get the results that are close to the real world phenomena. These steps are:

1. The real world phenomena is described by ideal scheme. Incorrect assumptions that were made in ideal scheme will make the results wrong.
2. The ideal scheme is approximated by differential or integral equations. Differential equations must correctly take into account the process components and their relations from ideal scheme. Some additional assumptions and simplifications can change the results making them non accurate or even incorrect.
3. Special numerical algorithms are applied to solve differential equations. These algorithms must be efficient and robust. Often it is important to use special algorithms depending on a model or model parameters.
4. Numerical algorithms are implemented in software in order to get the virtual object. The implementation itself can be inefficient and sometimes it is important to adapt the implementation to the specific hardware. The hardware related problems in this work are connected to parallel systems, in order to utilize this type of hardware it is critical to develop parallel algorithms. On this step original sequential algorithm often is modified in such a way that the new algorithm is equal to the original one in terms of results. However, additional computational error can appear because of a hardware-related reasons.

In this dissertation we have focused on points 3 and 4 while solving three modelling problems. The first problem is related to an efficient hardware usage when GPU is used to perform three-dimensional Finite-difference time-domain (3D FDTD) calculations. The second problem is about solving a complex mathematical problem with specific geometry, big variations of equation coefficients which can be generalized to model multiphysics processes – it is hard to solve it efficiently and we propose the efficient and robust solution, moreover, the problem is solved using parallel computing. The third problem is about modelling of bacteria self-organization process with formation of complex patterns when it is impossible to control the correctness of algorithms using classical mathematical modelling theory. We propose robust algorithms and their parallel versions and formulate the conclusions for future technology development in this direction. In all three problems parallel computing is used as an essential tool.

Problem Formulation

From mathematical point of view we use standard numerical methods: finite elements method (FEM), Finite volumes method (FVM) and Finite difference method (FDM). However, the usage of these methods may be inefficient and it is necessary to develop technological components that tune and optimize the software. The research involves three modelling problems with different specificities that make challenge to develop software to correctly solve these problems in efficient and robust way. There are three main scientific topics that are discussed in this work:

1. How to apply GPU technology efficiently to implement 3D Finite-difference time-domain methods.
2. How to apply Finite volume method efficiently to problems with complex geometrical domains and values of coefficients of materials may have big jumps on contours of different subdomains. Also in many cases it is important to provide parallel versions of solvers for clusters of computers with multiple nodes.
3. How to recognize and analyze the correctness of the modelling process, if we consider a problem that describes a process that forms a stochastic pattern (Eisenbach 2004).

Relevance of the Thesis

The usage of virtual experiment technology, when the physical experiment is changed by computational simulations, becomes more and more popular as computer technology increases it's potential. Better technology opens new possibilities – more complex models can be analyzed, better precision of results can be achieved. Computational possibilities are and (at least in the near future) will be the limiting factor for virtual experiments. Hence it is critically important to assure that computer resources are used efficiently. With more complex processes being considered the correctness of specific algorithms usage becomes more questionable. In order to efficiently utilize computational hardware, algorithms must be modified and adapted to computations on specific hardware.

Research Object

The object of research is distributed algorithms for physical phenomena modelling using Finite Volumes and Finite Differences methods.

The Aim of the Thesis

The main aim of this dissertation is to make the group of physical phenomena modelling problems solvable in an efficient and robust way using distributed computing algorithms.

The Objectives of the Thesis

Research objectives are split into 3 groups:

- Objectives for usage of GPU hardware for mathematical modelling with 3D FDTD method:
  1. Perform the analysis of possibilities of 3D FDTD implementation on GPU hardware.
  2. Propose an efficient approach for 3D finite difference time domain (3D FDTD) calculations on GPU using CUDA.
  3. Perform detailed analysis of 3D FDTD calculations on GPU. Create the prediction model for times of computations.
  4. Apply the results to a real physical problem.
- Objectives for efficient application of FVM to problems that have complex domain geometry and coefficients with big jumps:
  1. Perform the analysis of possibilities of FVM implementation for such problems.
  2. Perform calculations with a chosen FVM package applying it to the heat transfer modelling in high-voltage electrical cables. Analyze the quality of results, identify drawbacks and restrictions of solving approach linked to a specificity of a problem.
  3. Improve the quality of modelling process by modifying the package and/or by creating additional tools.

4. Investigate the efficiency and robustness of proposed approach, including the parallel version of computations.

- Objectives for modelling of unstable processes with complicated dynamics:
  1. Perform the modelling of bacteria self-organization (Eisenbach 2004), the process that has pattern forming instability.
  2. Analyze the possibility of distributed parallel calculations for such kind of problems, propose a general approach.
  3. Formulate conclusions and recommendations for making software for simulation of such kind of processes.

Research Methodology

The following methods were used to perform the research:

- Numerical schemes (FVM, FDM) for solving differential equations from mathematical modelling theory.
- Convergence analysis from theory of algorithms.
- Scalability analysis from theory of parallel computing.
- Regression analysis from theory of statistics.

Scientific Novelty of the Thesis

In the first part of the research an algorithm to solve the 3D FDTD problem with GPU using CUDA platform was proposed and analyzed. GPU was used widely before in this field and some recent researches (Wahl, Ly Gagnon, Debaes, Van Erps, Vermeulen, Miller, Thienpont 2013) show that this type of calculations can be automatically parallelized on several GPUs using special software packages. Other alternative could be the usage of implementation simplifying tools such as (Hoshino, Maruyama, Matsuoka, Takaki 2013), which in general has approximately 50% lower performance than CUDA, however, for some applications it can reach up to 98% of CUDA efficiency. It can be noted, that there is a possibility to implement calculations using OpenCL standard, however, some research by other authors shows that for NVIDIA GPUs

the usage of OpenCL is not more efficient than the usage of CUDA (Karimi, Dickson, Hamze (2010)).

In this work the CUDA platform is used as a base for implementation to make the research less dependent on effects of implementation specificity. The algorithm is based on work by Micikevicius (2009). The novelty of our research is that we perform more detailed scalability analysis and propose the model for prediction of computational time. Additionally we perform tests on different GPUs with different compute capabilities. Moreover, we show how to derive the algorithm from more general approaches, making it closer to a classical theory of parallel algorithms. Finally, we apply the proposed algorithm to a very specific numerical scheme that was created to solve a practical scientific problem (Šlekas 2014).

The second part of the research is dedicated to modelling of the heat transferring in electrical cables. It was done before (Karahan, Kalenderli 2011; Makhkamova 2011), however, in this work the main focus is made on the efficiency, that helps to create software efficiently utilizing the computer resources. Acute triangles and Voronoi points were used to achieve these results. The adaptive mesh was constructed and a new original algorithm to control mesh singularities was proposed. Parallel version of the algorithm is generated automatically using OpenFOAM, however, in the research it is shown what parameters (that should be selected from the big list provided by OpenFOAM) and how they should be used in order to perform calculations efficiently. Scalability analysis is performed. Research was done in context of creating a software for modelling of heat transferring in electrical cables, however, the results can be applied to a wide range of other similar problems.

In the third part we have shown that chemotaxis-driven instability can be connected to the ill-posed problem defined by the backward in time diffusion process. The problem was approximated by the discrete computational model using the well-established techniques. Results of numerical experiments show that for such problems, where chemotaxis-driven instability defines the dynamics of a solution, the classical convergence property of numerical algorithms is not applicable. Instead of pointwise and similar convergence metrics we should use qualitative criteria (e.g. the theory of attractors) or averaged statistical characteristics (as, e.g., in the discrete element method). The parallel algorithm version is proposed and the results of scalability analysis are provided.

Practical Value of Research Findings

3D FDTD method is widely used to solve Maxwell's equations. For software development it is necessary to analyze the efficiency of implementation on different GPUs. Computations time prediction model lets to evaluate GPUs in terms of usefulness of using GPU for 3D FDTD computations. There are plenty of researches that are done using Tesla GPUs that are mainly oriented for GPU computing. However, for software engineers it is more important to evaluate GPU usage for average customer on GPU market with typical real world applications (so called general-purpose graphics processing units (GPGPU) users).

The efficient modelling of heat transferring in electrical cables is essential to solve industrial optimization problems. The proposed technological solutions can be used for development of a software aimed to optimize cable design, network infrastructure of power transmission lines, the usage of existing infrastructure. Some problems require enormous amount of computations – it can take weeks of calculations even using supercomputers. At present the power lines and cable design are identified as important direction for optimization. Moreover, the results can be easily applied to a wide range of other similar problems, it is not limited to the modelling of heat transferring in cables only. Tests of parallel versions of solvers showed that the suggested solvers are efficient on multi-core and even multi-CPU systems.

The development of efficient and robust technological solution is essential when the model of physical process is unstable in terms of classical convergence and data sensitivity criteria from mathematical modelling theory. All standard numerical methods for solving differential equations were studied using this theory and the convergence was proved using classical error norms – that is why we need to study this case more deeply, before we can implement the methods into a software. In this dissertation we have analyzed how to simulate the situation when the model forms stochastic patterns. It is essential for creating a software to model such problems, since the methods must assure that instability(sensitivity) is consequence of a model, not of the methods. Moreover, we propose a parallel algorithm version, in case if it is needed to speedup calculations using parallel computing.

Defended Statements

1. The algorithm for GPU usage for stencil-type calculations (more specifically – 3D FDTD method) analyzed, proposed time prediction model predicts the main order of 3D FDTD (Šlekas 2014) computational time of GPU solver, i.e. the prediction and actual computational times do not differ more than 10 times. Application of this GPU algorithm for 3D FDTD tested cases gives a reasonable calculations time speedup comparing to CPU version.
2. The proposed approach to model specific set of heat transferring problems is efficient and robust, efficiently utilizes computer resources when parallel clusters of computers are used, thus it fits to be used for optimization software development.
3. A specific non-stable process forms complex patterns due to model, thus it is important to chose stable numerical methods to ensure they have no qualitative effect on pattern formation. Proposed robust algorithm models the process correctly, however, special metrics (norms) to validate the model and its parameters itself are needed.

Approval of Research Findings

The results of the research were presented in 6 international and 3 national Lithuanian conferences. 4 articles were published in periodical scientific publications in journals referred in ISI Web of science. 3 articles were published in publications that are referred in ISI Conference Proceedings. 1 article was published in journal referred in other database. The detailed list of publications can be found in the “List of Scientific Publications by the Author on the Topic of the Dissertation” theme.

Structure of the Dissertation

Dissertation consists of Introduction, 3 chapters, conclusions, bibliography, appendixes. Chapters are divided into sections, sections – into subsections. The numeration used in dissertation is “chapter.section.subsection”, “chapter.figure”, “chapter.table”. The numeration of formulas is separate for each chapter, “chapter.formula”.

The dissertation has 129 pages without appendixes, with 201 formulas, 53 figures and 20 tables. In dissertation it was cited 106 of other authors publications.

Aknowledgement

I am grateful to my supervisor Prof Dr Habil Raimondas Čiegis for supervising my work. His leadership and support during my PhD studies have created conditions that are close to optimal for the growth of my scientific competence, which is the main goal of such studies.

I want to thank to my colleagues who have helped to perform my research. A special thanks is given to Dr Gediminas Šlekas and Prof Dr Habil Žilvinas Kancleris for cooperation in performing the research and providing some important additional computational resources.

Also, I want to thank to reviewers Prof Dr Olegas Vasilecas and Prof Dr Zenonas Navickas who helped to make valuable corrections to the dissertation, and to head of department for doctoral studies Assoc Prof Dr Šarūnas Mikaliūnas for the help in preparing the text of dissertation.

The work was supported by Eureka project E!6799 POWEROPT “Mathematical modelling and optimization of electrical power cables for an improvement of their design rules”.

1

3D Finite-difference Time-domain Calculations Using Graphics Processor Unit

In this chapter it will be discussed how to apply GPU technology efficiently to implement 3D Finite-difference time-domain methods. An algorithm to solve the 3D FDTD problem with GPU using CUDA platform was proposed and analyzed. The algorithm is based on work by Micikevicius (2009), but here will be shown how to derive the algorithm from more general approaches, making it closer to a classical theory of parallel algorithms. The algorithm was implemented using CUDA and applied for 3D FDTD calculations. The implementation was proved to be efficient in terms of utilizing GPU resources. It gave a reasonably fast computations times. The detailed computations analysis was performed on three different GPUs, the model for prediction of computations time was proposed. This model lets to evaluate different GPUs in terms of their efficiency for 3D FDTD calculations. However, the model should be modified to be more general so it could be applied to a newer GPUs. The real techno-

logical problem was solved using this implementation, results were presented in another dissertation (Šlekas 2014).

Parts of this chapter are published in (Čiegis, Bugajev, Kancleris, Šlekas 2014).

1.1. Introduction Into First Chapter

Parallel computing is a part of a mainstream in computational science, and the main condition for that is availability of computational hardware technology (Herlihy, Luchangco 2008; Marowka 2007; Muhammad, Eberl 2010; Pankratius, Tichy 2008; Thomaszewski, Pabst, Blochinger 2008): multicore processors, multi-processor desktop computers, clusters of computers and more recently also general purpose computing graphics processing units (GPU). The last one which is a relatively new technology that have created new challenges for computational mathematics and programmers in the field of parallel computing. GPU, which is inside of the most new video cards, is capable of performing huge numbers of FLOPS (floating point per second). In certain circumstances new GPUs overcome any new CPU in terms of performance/price. However, GPUs have many restrictions and requires specific algorithms to fit these restrictions and to use GPUs resources efficiently. The most important task of calculations on GPU is the optimal implementation of data transfers to computing unit. That is why the main GPU computation efficiency indicator is GPU bandwidth (e.g. number of elements per second, or GB/s (gigabytes per second)). The bandwidth obtained in application compared to peak bandwidth shows how effectively GPU is used. To use GPU for calculations by NVIDIA suggests CUDA platform. Solving Maxwell equations on CUDA platform with FDTD method was studied in many papers. For example (Adams, Payne, Boppana 2007) and (Liuge, Kang, Fanmin 2009) both showed good speed-ups comparing to traditional CPU computations. Micikevicius (2009) has implemented 3D FDTD, this work is included in CUDA SDK as an example project. Wahl, Ly Gagnon, Debaes, Van Erps, Vermeulen, Miller, Thienpont (2013) shows that this type of calculations can be automatically parallelized on several GPUs using special software packages. Eberl, Sudarsan (2014) have used an accelerator programming standard OpenACC, that simplifies the usage of GPU, to parallelize diffusion problems, applied to temperature distribution on a honeycomb around the bee brood. Hoshino, Maruyama, Matsuoka, Takaki (2013) has investigated OpenACC, which in general has approximately 50% lower performance than CUDA, however, for

some applications it can reach up to 98% of CUDA efficiency. It can be noted, that there is a possibility to implement calculations using OpenCL standard, however, some research by other authors shows that for NVIDIA GPUs the usage of OpenCL is not more efficient than the usage of CUDA (Karimi, Dickson, Hamze (2010)). In this dissertation the CUDA platform by NVIDIA is used as a base for implementation to make the research less dependent on effects of implementation specificity.

The algorithm used in this dissertation is based on work by Micikevicius (2009). However Micikevicius has tested the implementation on Tesla GPU and does not give expected speed-ups on GPUs that are used in this research. Moreover, Micikevicius do not perform scalability analysis for different calculation templates and different GPUs, this information is very important for informatics engineers that are oriented to average consumer on the computers market. That is why the independent research is performed in this work, with analysis of impact of different calculations specificities and GPUs on efficiency. Lots of experimental data is presented by different researches in this field. However, there is a lack of mathematical formalism that would make analysis of calculations on GPU technology closer to the classical theory of parallel algorithms. Our goal is to systematize the GPU hardware engineering facts and specific properties of the parallel algorithms developed for data parallel computations into one computational model. Such a template can help to predict the complexity of computations and enable the user to make a scalability analysis of the developed parallel algorithms. Then the performance of the algorithm can be optimized with respect to the given parameters of the model (Starikovičius, Čiegis, Iliev 2011).

Compute Unified Device Architecture

All the technical details that are collected in this section were taken from the official NVIDIA documents “NVIDIA CUDA C Programming Guide” and “CUDA C Best Practices Guide”.

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model that is orientated to NVIDIA GPUs (graphics processing unit). CUDA architecture is strongly relative to GPUs architecture. CUDA as programming model includes only the most important (in terms of computation) aspects of GPUs architecture, hiding from programmer unneeded device hardware details. That is why in this work it will not be discussed about GPUs micro architecture with all specific aspects of it. Generally the whole architecture can be separated to

1. Host processor – CPU processor, standard computing unit with host memory (RAM), it is not a part of GPU, however, GPU need to communicate with host (CPU).
2. Device, i.e. GPU itself.

The device consists of array of stream multiprocessors (SMs). They execute blocks of threads. In general one SMs can execute more than one block of threads simultaneously. There are some important restrictions for using SMs:

- the number of active blocks per SM is limited,
- blocks of threads are distributed on SMs as is shown on Fig. 1.1,
- during program execution, all threads are grouped by up to 32 threads, called warps,
- the number of active warps (summing all warps from blocks that run simultaneously) per SM is limited,
- the number of threads per block is also limited and this number is smaller than the maximum number of threads per SM.

The structure of CUDA memory space is shown on Fig. 1.2. Device execution memory is described by the following hierarchy:

- host memory, which is not actually the part of device, but device must transfer data from/to host,
- global memory, which is accessible by threads that run on SMs,
- shared memory, which is located on each SM and is accessible only by this SM. It is much faster than global, but it has a very limited size. This memory is shared between all threads in one block. The size of memory is limited per SM, it is not directly accessible by host.

There are also the other three types of memory: constant, texture and local. Constant memory is used for read-only constants and is kept on GPU cache. The texture memory is also read-only and is placed on GPU cache. The local memory actually is not a separate memory space, it is located on the global memory and is reserved for separate threads in cases when there is not enough registry during execution.

Let us overview the most important steps for any algorithm implemented on CUDA.

1. Task parallelization:
   - the task is split into blocks of SIMD tasks (Single Instruction Mul-

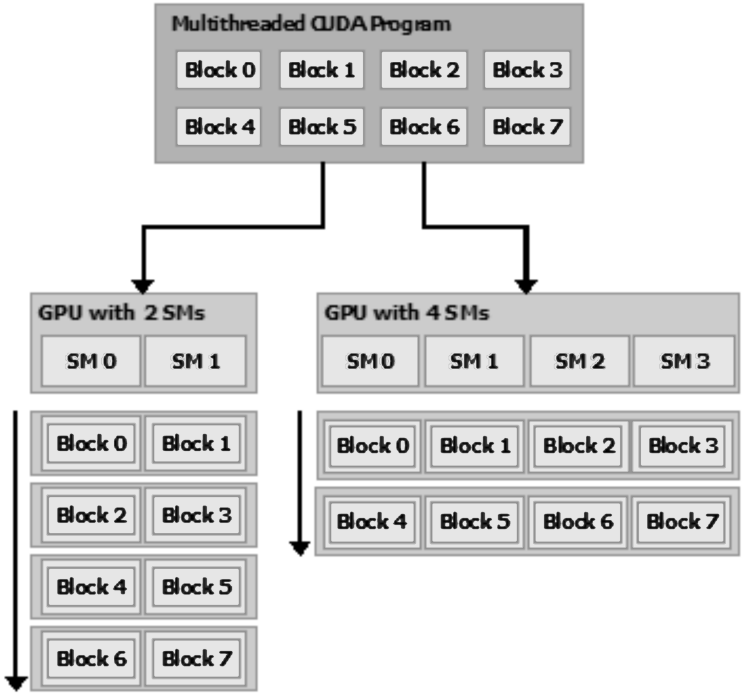


Fig. 1.1. Distribution of thread blocks on streaming multiprocessor

tiple Data), which will be performed by different threads in parallel.

- Array of blocks is constructed, and each block is indexed. Indexation can be one-dimensional, two-dimensional or three-dimensional,
- tasks in a block are assigned to threads, which are indexed inside a block,
- in programming process the task is identified by block and task indexes,
- all GPU operations are performed by warps (32 threads) or half-warps (16 threads). Thus it is important to have portions of 32 tasks (or 16 for older GPUs) in block of tasks, so all threads in a warp could execute their tasks,
- Data transfers are performed by warp transactions and must satisfy data alignment requirements to be performed effectively, these re-

quirements vary on different GPUs with different compute capabilities.

2. Let us note the main aspects which are important for threads to be running simultaneously: a) Each SM executes blocks of tasks that are given from array of blocks. b) Each SM can simultaneously execute a limited number of thread blocks which is limited by the following restrictions: there is a maximum number of active warps per SM N_{wmax}, a maximum number of registers N_{rmax}, a maximum size of shared memory per SM and a maximum number of active blocks per SM N_{bmax}.

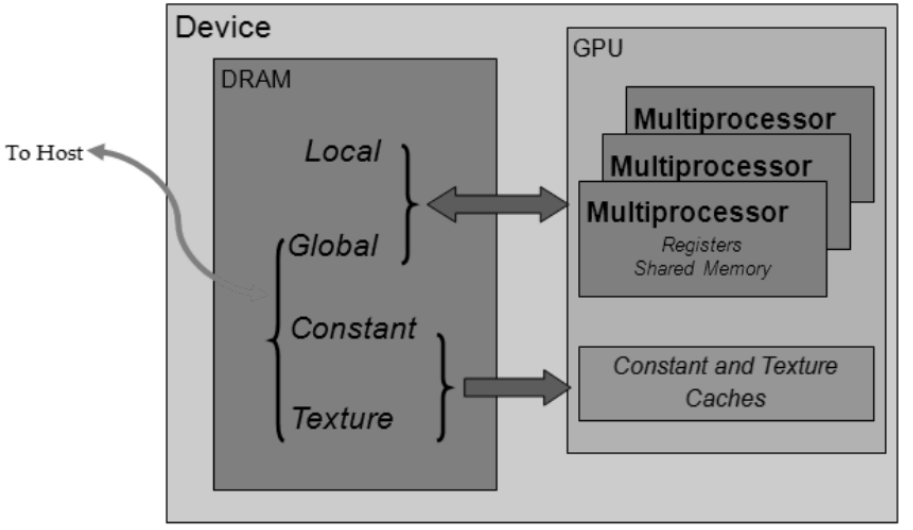


Fig. 1.2. Memory spaces on Compute Unified Device Architecture device

1.2. Occupancy

The GPU occupancy is defined as the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Note that executing other warps when one warp is paused or stalled is the only way to hide latencies and keep the hardware busy. Higher occupancy does not always equate to higher performance, often there is a point above which additional occupancy does not improve performance or it has negligible effect.

However, low occupancy always interferes with the ability to hide memory latency, resulting in performance degradation.

Let N_t be the number of threads per block, N_b the number of active blocks per SM, N_r the number of registers per thread used, N_s the size of shared memory per block, N_b the number active blocks per SM, N_{tmax} the maximal number of threads per SM, N_{smax} the total shared memory (in floats), N_{tbmax} the maximal number of threads per block. Then the main restrictions for GPU resources can be written as

$$\begin{cases} N_t \le N_{tbmax}, \quad N_b \le N_{bmax}, \quad N_t N_b \le N_{tmax}, \\ N_b N_s \le N_{smax}, \quad N_b N_t N_r \le N_{rmax}. \end{cases} \tag{1.1}$$

Assume that each active warp is using all 32 threads, then the occupancy ϱ is calculated as

$$\varrho = N_t N_b / N_{tmax}. \tag{1.2}$$

It is easy to see that $\varrho \in (0, 1]$. In terms of affects for occupancy, any implementation of an algorithm can be described by parameters N_t, N_s and N_r. From (1.1) the maximum number of active blocks per SM should satisfy conditions

$$N_b = \min\left(N_{bmax}, \left[\frac{N_{tmax}}{N_t}\right], \left[\frac{N_{smax}}{N_s}\right], \left[\frac{N_{rmax}}{(N_t N_r)}\right]\right), \tag{1.3}$$

where $[\cdot]$ is the integer part of real number. After we have parameters N_b and N_t, ϱ is calculated from (1.2). Let us discuss how to obtain the maximum occupancy value $\varrho = 1$. Note that N_{tbmax} usually satisfies the condition $N_{tbmax} < N_{tmax} \le 2N_{tbmax}$, then to get $\varrho = 1$ (when all threads of SM are active) it is needed to have at least two active blocks $2 \le N_b \le N_{bmax}$. (1.2) and estimates above give

$$N_t \le N_{tmax}/2, \quad N_b = N_{tmax}/N_t. \tag{1.4}$$

Conditions (1.1), (1.4) give the the required estimates

$$\frac{N_{tmax}}{N_{bmax}} \le N_t \le \frac{N_{tmax}}{2}, \quad N_s \le N_t \frac{N_{smax}}{N_{tmax}}, \quad N_r \le \frac{N_{rmax}}{N_{tmax}}. \tag{1.5}$$

Assuming that $\dfrac{N_{tmax}}{N_t} = \left[\dfrac{N_{tmax}}{N_t}\right]$, (1.5) are necessary and sufficient conditions for all threads to be active, i.e. $\varrho = 1$. Note, that if the occupancy

ϱ is already close to 1, then an attempt to increase ϱ will not give a reasonable speed-up for calculations. But small values of ϱ make threads to be idle, blocking the possibility to use reserved GPU resources.

Originally in NVIDIA documentation the description of occupancy is oriented to users that are not necessary scientists and is done with lack of formalism and formulas that would let to make it much shorter. In this section the occupancy was formalized and conditions for optimal occupancy were formulated.

Lets apply these results to an example, when optimal parameters for calculations are chosen. Calculations are performed using 32 bit float data type on GeForce 9400 GT video card with theoretical peak global memory bandwidth equal to 12.8 GB per second ($\beta_2 \geq 4/(12.8 \cdot 10^9)$). GeForce 9400 GT have 2 SMs with these parameters (they are obtained by calling CUDA special functions): $N_{tmax} = 768$, $N_{bmax} = 8$, $N_{tbmax} = 512$, $R_{rmax} = 8192$(4 bytes each, or one float), $R_{smax} = 16384(bytes) = 16384/4(floats) = 4096floats$. Calculations are performed by $N_{tx} \times N_{ty}$ sized blocks, so $N_t = N_{tx}N_{ty}$. Three arrays are kept in shared memory, so $N_s = 3N_t$. Next, tests to estimate how well does model (1.13) fit calculations will be performed. Calculations with number of register per thread $N_r = 11$ were implemented. So (1.3) gives

$$N_b = \min\left(8, \left[\frac{768}{N_t}\right], \left[\frac{4096}{3N_t}\right], \left[\frac{8192}{11N_t}\right]\right) = \min\left(8, \left[\frac{8192}{11N_t}\right]\right), \quad (1.6)$$

so ϱ is limited by number of registers if $N_b = \left[\frac{8192}{11N_t}\right] \leq 8$, (1.2) gives

$$\varrho = \frac{N_t \cdot \left[\frac{8192}{11N_t}\right]}{N_{tmax}} \leq \frac{736}{768} = 0.875, \quad (1.7)$$

since N_t must be multiple of 16 (half-warp for occupancy 1.1 GPU), ϱ obtains it is maximum value when $N_t = 96$, $\varrho = 0.875$, $N_b = 7$.

In all computational tests optimal values are used.

1.3. The Template of Numerical Algorithms

A typical finite difference or finite volume approximation of the 3D FDTD problem can be represented by sequential calculations of solu-

tions of very large sparse systems of linear equations at each time step. One iteration can be described in the following compact form $M = F(A_1, A_2, \ldots, A_m)$, where $M, A_1, A_2, \ldots, A_m$ are $N_x \times N_y \times N_z$ dimension arrays and operator F is defined on the given stencil of the grid $M_{ijk} = f(S_{ijk}(A_1), S_{ijk}(A_2), \ldots, S_{ijk}(A_m))$, where $S_{ijk}(A)$ is a set of elements around the element $A_{i,j,k}$, f is a problem dependent function. The set $S_{ijk}(A)$ is described by a stencil of the approximation algorithm. For example, let us consider a cross stencil with radius equal to $s = 1$, then S_{ijk} is defined as:

$$S_{i,j,k}(A) = \left\{A_{ijk}, A_{i-1,j,k}, A_{i+1,j,k}, A_{i,j-1,k}, A_{i,j+1,k}, A_{i,j,k-1}, A_{i,j,k+1}\right\}. \tag{1.8}$$

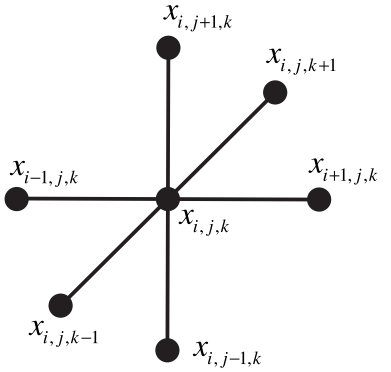


Fig. 1.3. The cross stencil

Elements (1.8) describe data which is needed for calculations (see Fig. 1.3). To parallelize calculations it is necessary to split the corresponding data into blocks and update information separately in parallel. Since shared memory on a device has a very limited size, these blocks should be small enough.

When the data is split into blocks, each block must be expanded by additional elements called *halo* elements (Micikevicius 2009). To indicate the efficiency of data transfer for different implementations, define *redundancy* R as the ratio between the number of elements accessed and the number of elements processed. The redundancy $R = R_1 + R_2$ consists of read redundancy R_1 and write redundancy R_2. It is assumed that the number of elements read is N_r, the number of elements written is N_w, the number of elements processed

is N_p, then the redundancy is calculated by the following formula:

$$R = N_r/N_p + N_w/N_p.$$

In ideal case the numbers of elements read and written are equal to the number of elements processed, then the read and write redundancies both are equal to 1 and the overall redundancy is equal to 2. However, because of a stencil halo elements are needed to be read, so the redundancy is bigger than two for a stencil-driven calculations.

2D data splitting. The 3D processed data is split in x and y directions into blocks with sizes $N_{tx} \times N_{ty} \times N_z$. Since the whole block can not be placed into a shared memory, it is needed need to construct an algorithm for data to be processed in the right order. So the whole block is split into subblocks with sizes $N_{tx} \times N_{ty} \times N_{tz}$. Since calculations involve the information on a stencil set, then to process the subblock of data additional elements from each side of subblock needed to be accessed. The total number of additional elements is equal to $2(N_{tx}N_{ty} + N_{ty}N_{tz} + N_{tx}N_{tz})s$, where s is the radius of the stencil.

All blocks and subblocks are equal in terms of calculations and data transfers, so the behavior of one block determines the read and write redundancy for the given algorithm. For each subblock $N_{tx}N_{ty}N_{tz}$ elements are processed and written, but $N_{tx}N_{ty}N_{tz} + 2s(N_{tx}N_{ty} + N_{ty}N_{tz} + N_{tx}N_{tz})$ elements are read, so the overall redundancy is

$$R = 2 + 2s(N_{tx}N_{ty} + N_{ty}N_{tz} + N_{tx}N_{tz})/(N_{tx}N_{ty}N_{tz}). \qquad (1.9)$$

Note, that in this case, when subblocks are processed in one direction, a part of data from the previous subblock can be kept. There will be a reference to this modification later.

3D data splitting. In 3D data splitting case it is not necessary to process the data by subblocks, since the appropriate sizes can be chosen $N_{tx} \times N_{ty} \times N_{tz}$ of subblocks and the redundancy is the same as in the 2D splitting case. However, there is no possibility to reuse data from the previous block when some new block is processed.

The modified 2D data splitting. As was mentioned above, in 2D splitting case we can keep a part of data of one subblock to update the other subblock. To be exact, the neighbour subblock in z direction needs $2sN_{tx}N_{ty}$ elements

of the current subblock. So this subblock needs to read only $N_{tx}N_{ty}N_{tz} + 2s(N_{tx}N_{ty} + N_{ty}N_{tz} + N_{tx}N_{tz}) - 2sN_{tx}N_{ty} = N_{tx}N_{ty}N_{tz} + 2s(N_{ty}N_{tz} + N_{tx}N_{tz})$ elements. The overall redundancy is

$$R = \frac{2N_{tx}N_{ty}N_{tz} + 2s(N_{ty}N_{tz} + N_{tx}N_{tz})}{N_{tx}N_{ty}N_{tz}} = 2 + \frac{2s(N_{ty} + N_{tx})}{N_{tx}N_{ty}}. \quad (1.10)$$

It can be seen from (1.10) that the redundancy does not depend on N_{tz}, so it is optimal to take $N_{tz} = 1$ in order to use as small amount of shared memory as possible. The equivalent aim is to make N_{tx} and N_{ty} as big as possible in order to get smaller redundancy (1.10). So the subblock becomes a slice in x and y plane. Then we have that $N_{tz} = 1 \ll N_z$, so iteratively reading subblocks we can ignore the first subblocks redundancy (1.9), and assume that all subblocks has redundancy (1.10).

This approach is already mentioned by Micikevicius (2009). However, he interpreted this approach in a different way – as two-pass approach, similar to one described for Cell processors (see ((Micikevicius 2009)) for more details). This technique in this work was obtained from general data splitting paradigms of parallel computations theory.

Let us conclude the results obtained from 2D splitting. We split all 3D data into 2D blocks with sizes $N_{tx} \times N_{ty} \times (N_z - 2s)$. Then the size of grid of blocks is $\frac{(N_x - 2s)}{N_{tx}} \times \frac{(N_y - 2s)}{N_{ty}} \times 1$. In each block, the data is processed iteratively by slices in z direction. The pseudo code for calculations is presented in Algorithm 1.

Algorithm 1 The pseudo code for calculations

1: **for** $k = s \to N_z - s - 1$ **do**
2: calculate $M_{i,j,k}\, for\, \forall i, j$ inside the block
3: **end for**

Note, that in the cross stencil case halo elements in z direction can be stored in the registry memory, since they are needed for one thread only. At the first step of one block calculations, the data is copied only for the first slice. The pseudo code of this algorithm for one block is given in Algorithm 2.

Algorithm 2 The pseudo code for one block calculations

for $i = s \rightarrow N_z - s - 1$ **do**
  if $i == s$ **then**
    $sN_{tx}N_{ty}$ elements of slices $z = 0, \ldots, s-1$ and $z = s+1, \ldots, 2s-1$
    are copied from device to registry memory;
    $N_{tx}N_{ty} + 2s(N_{tx} + N_{ty})$ elements (with halo) of slice with $z = s$
    are copied from device to shared memory;
  else
    1. the slice with index $(i - s - 1)$ is deleted from registry memory;
    2. the elements (without halo) from slice with index $i-1$ are copied
    from the shared to the registry memory, then the shared memory
    used for these elements is freed;
    3. shared halo elements from slice with index $(i - 1)$ are deleted
    from shared memory and halo elements from slice
    with index i are copied to shared memory from device;
    4. the elements from slice with index i are copied from registry
    to shared memory;
  end if
  5. the slice with index $(i + s)$ copied from device to registry memory;
  6. the data from slice with $z = i$ is processed and written back
  to device;
end for

1.4. Complexity Analysis

Consider a mathematical model of the complexity of the given algorithm:

$$T = \alpha + \beta N + \gamma \bar{N}, \tag{1.11}$$

where T is computation time for the implementation of one iteration of the algorithm, α is time taken to prepare data for transfers, β determines time required to send one element, γ determines how much time units are needed to perform one basic operation, $\bar{N}$ is the number of operations.

In CUDA there are two main transfer operations: from host to global memory and from global memory to shared/registry/cache. Assuming that start-up

costs are negligible $\alpha = 0$, we modify the model (1.11):

$$T = \beta_1 N_1 + \beta_2 N_2 + \gamma \bar{N}, \tag{1.12}$$

where β_1 and β_2 determine time required to send one element from/to host to/from global memory and from/to global memory to/from shared/registry respectively.

For algorithms considered in this section the cost of data transfers between global and shared memory is the most important. So in the theoretical model we consider transfer from/to shared memory and computation times only:

$$T = \beta_2 N_2 + \gamma \bar{N}. \tag{1.13}$$

The parameter β_2 depends on many factors such as: GPU occupancy ϱ, redundancy R and some more specific aspects of algorithm implementation. R shows how many transfers are made per one element processed, ϱ shows how effectively the hardware is used. So from the algorithmic point of view we would assume that

$$\beta_2(\varrho, R) = R/W(\varrho),$$

where W is bandwidth, a monotonically increasing function, that is determined experimentally.However for calculations with CUDA many other details become more important than R, and we propose the following simple formula for β_2 calculations

$$\beta_2 = (1 + \beta_h d_h)/W(\varrho), \tag{1.14}$$

where β_h is experimentally determined constant, that shows the difference between transfer times of halo and regular elements

$$d_h = \frac{\text{number of transfers of arrays with halo elements}}{\text{total number of transfers of arrays}}.$$

Note that (1.14) does not depend on R, so it makes no sense in terms of choosing optimal block size. Suppose, that bandwidth drops due to additional stalls of threads at synchronization points, that are necessary to perform shared memory management for using by the whole block of threads.

In order to determine the function $W(\varrho)$ we consider a simple benchmark with three input arrays A, B, C, and one output array D. One-point grid stencil $S_{i,j,k}(A) = A_{i,j,k}$ is used, so there are no halo elements. We take the kernel function F which is defined by the pseudo-code in Algorithm 3.

Algorithm 3 The function for benchmark to measure $W(\varrho)$

```
f(A_{i,j,k}, B_{i,j,k}, C_{i,j,k}) :
Begin
R = 1
for i = 1 → K do
    D = D + D * A_{i,j,k}
    D = D + D * B_{i,j,k}
    D = D + D * C_{i,j,k}
end for
End
```

Here K is the number of iterations that we use as a simple floating point per second multiplicator, so the number of operations (floating points) is $6K$ per element calculated. In computational experiments we have used 3 different video cards, we will refer to them as GPU1, GPU2, GPU3:

- GPU1: GeForce 9400 GT, compute capability 1.1, 12.8 GB/s, 67.2 GFlops
- GPU2: GeForce GTX 460, compute capability 2.1, 128 GB/s, 1042 GFlops
- GPU3: GeForce GTX 650 Ti, compute capability 3.0, 86.4 GB/s, 1425 GFlops

In Table 1.1, the maximum values of GFlops and bandwidth (Gbytes/s) that were achieved for this benchmark are presented, they are compared to theoretical peak of corresponding values ("t.peak" in table). Note that it can

Table 1.1. Implementation efficiency analysis

GPU	bandwidth (GB/s)	t.peak	GFlops	t.peak
GPU1	11.1	12.8	49.5	67
GPU2	94.6	128	583	874
GPU3	59.5	86.4	916	1425

be achieved more GFlops, but it is not important, since usually β_2 has bigger impact on efficiency calculations. The GPU parameters are presented in Table 1.2.

Then model (1.13) can be rewritten as

$$T = \frac{4N}{W(\varrho)} + 6K\gamma N, \tag{1.15}$$

Table 1.2. The model parameters for GPUs

GPU	β_0	β_h	γ
GPU1	4.22E-10	7.6	19.4E-12
GPU2	4.88E-11	3.27	1.72E-12
GPU3	9.26E-11	–	0.923E-12

here $N = N_x N_y N_z$. γ is determined by fixing ϱ and changing K. After that K is fixed and ϱ is changed to determine $W(\varrho)$. In all the following computational experiments arrays sized $256 \times 256 \times 256$ are used, the number of time steps is 10. Increasing these values have a small effect on the results that we are interested in. ϱ can be controlled by allocating fictive shared memory arrays, that limits the maximum number of active thread blocks per multiprocessor. Assuming that ϱ is limited by shared memory, then (1.2) and (1.3) gives the occupancy value

$$\varrho = N_t \left[\frac{N_{smax}}{N_s} \right] / N_{tmax}.$$

Thus by allocating different-sized shared memory arrays ϱ can be controlled by choosing it from the range

$$\varrho = N_t/N_{tmax},\ \ 2N_t/N_{tmax},\ \ \ldots,\ \ 1.$$

$\varrho = \varrho_0$ is fixed and K is being changed in model (1.15) to obtain the results of Fig. 1.4 and Fig. 1.5. After λ is obtained, function $W(\varrho)$ is determined experimentally, by taking $K = 1$.

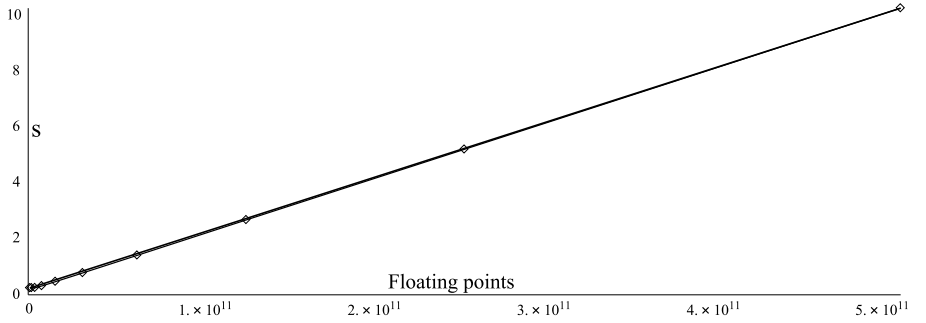


Fig. 1.4. GPU1 times with different number of floating points. The line with dots represents empirical data, the thicker line represents model $T(\bar{N})$

To analyze independence of β_2 on block sizes we take different sizes of blocks of threads in x and y directions d_x, d_y by changing sizes of CUDA

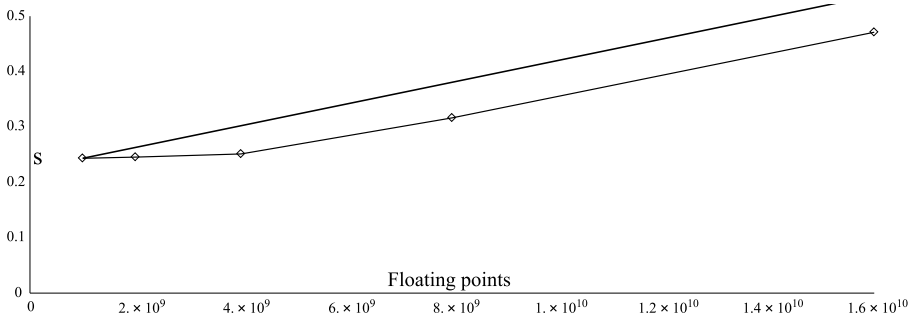


Fig. 1.5. GPU1 times with different number of floating points, zoomed version. The line with dots represents empirical data, the thicker line represents model $T(\bar{N})$

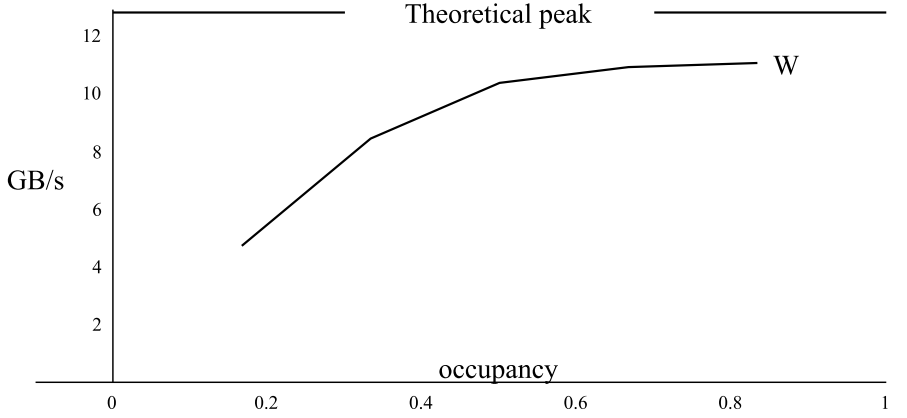


Fig. 1.6. GPU1 bandwidth $W(\varrho)$ measured in Gbytes/s

blocks of threads. As a benchmark for each iteration we transfer three arrays with halo elements from the global to the shared memory and one simple array back to the global memory. Results are presented in Table 1.3.

As it is seen from the table the times are roughly the same for different sizes of block of threads. However, this does not apply to GPU3 card – in this case β_h must be determined for each fixed block size. For GPU3 the smaller block sizes give better times, our guess is that GPU3 manages to synchronize smaller blocks better, since there are less threads for synchronization.

From our model we have $\beta_h = \dfrac{TW(\varrho) - N}{d_h N}$. We present experimental values of parameters β_0, β_h, γ for different GPUs, here $\beta_0 = 1/W(1/3)$:

The value of β_h for GPU3 depends on block sizes, however it is similar to

Table 1.3. CPU times with different sizes of thread blocks on GPU2

d_x	$d_y = 32$	$d_y = 64$	$d_y = 128$
1	–	0.090342	0.0883086
2	0.0902548	0.0857643	0.0838882
4	0.0914037	0.0876535	–
8	0.101497	–	–

Table 1.4. The model parameters for our GPU's

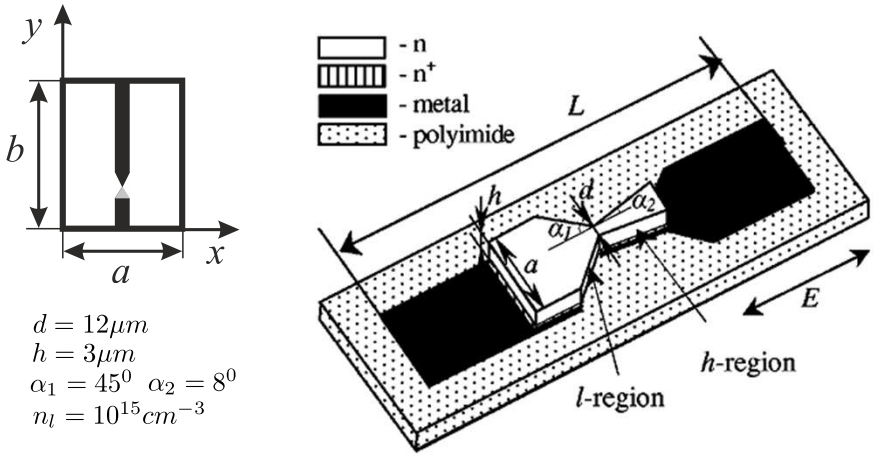
GPU	β_0	β_h	γ
GPU1	4.22E-10	7.6	19.4E-12
GPU2	4.88E-11	3.27	1.72E-12
GPU3	9.26E-11	–	0.923E-12

GPU2 value, that means the assumption for β_h to be constant in GPU3 case is invalid, and it needs more general model which is out of scope of this research. β is much bigger than γ that means if $\bar{N}$ and N have the values of the same order (what is usually true) then β has the biggest impact on calculations time. In other words the bottleneck for calculations is usually memory bandwidth. So if β has much bigger impact on calculations time than γ, then the complexity model predicts the smallest computation times when GPU2 is used. At this point the comparison between different GPUs is finished. In all remaining tests for simplificty only GPU2 will be used.

1.5. Application to a Real Problem – Propagation of TE_{10} Wave in Rectangular Waveguide

Consider the task of creating the planar diode which is considered by Šlekas (2014). Schematically the diode is presented on Fig. 1.7. To construct this diode some optimization tasks must be solved (see (Šlekas 2014) for details). In order to do that we need to model a wave propagation in rectangular waveguide (Fig. 1.8). Next we consider the numerical algorithm which was used to solve this modelling problem.

The details of the numerical algorithm are presented by Kancleris, Šlekas, Čiegis (2013). In terms of developing the solver for calculations on GPU the main part of one time step of calculations is sufficient. The main part of one



$d = 12\mu m$
$h = 3\mu m$
$\alpha_1 = 45^0 \;\; \alpha_2 = 8^0$
$n_l = 10^{15} cm^{-3}$

Fig. 1.7. Planar diode

time step of calculations can be represented in a form

$$
\begin{aligned}
H_{x,i,j,k} &+= 0.5(E_{y,i,j,k+1} - E_{y,i,j,k}) - 0.5(E_{z,i,j+1,k} - E_{z,i,j,k}),\\
H_{y,i,j,k} &+= 0.5(E_{z,i+1,j,k} - E_{z,i,j,k}) - 0.5(E_{x,i,j,k+1} - E_{x,i,j,k}),\\
H_{z,i,j,k} &+= 0.5(E_{x,i,j+1,k} - E_{x,i,j,k}) - 0.5(E_{y,i+1,j,k} - E_{y,i,j,k}),\\
E_{x,i,j,k} &+= 0.5(H_{z,i,j,k} - H_{z,i,j-1,k}) - 0.5(H_{y,i,j,k} - H_{y,i,j,k-1})\\
E_{y,i,j,k} &+= 0.5(H_{x,i,j,k} - H_{x,i,j,k-1}) - 0.5(H_{z,i,j,k} - H_{z,i-1,j,k})\\
E_{z,i,j,k} &+= 0.5(H_{y,i,j,k} - H_{y,i-1,j,k}) - 0.5(H_{x,i,j,k} - H_{x,i,j-1,k}),
\end{aligned}
$$

here all terms except the constants are 3D arrays. From calculations scheme we get:

- $N = 256^3 \cdot 10 \cdot 18$, here 256 is the sizes of arrays, 10 is the number of time steps. 18 is the number of arrays that must be transferred to/from shared memory: to calculate H arrays GPU needs to transfer 3 E and 3 H arrays to shared memory and 3 H arrays from shared memory, meaning 9 array transfers, the same applies for calculations of E arrays that gives $9 + 9 = 18$ arrays in total.
- $\bar{N} = 256^3 \cdot 10 \cdot 36$, since there is 6 operations in each line of the algorithm, 6 such lines total results in $6 \cdot 6 = 36$ operations.
- $d_h = 1/3$. E must be calculated after calculations of H are finished so the whole calculations are separated into two phases: for E and for H. At each phase operation $+=$ results in 2 transfers without halo

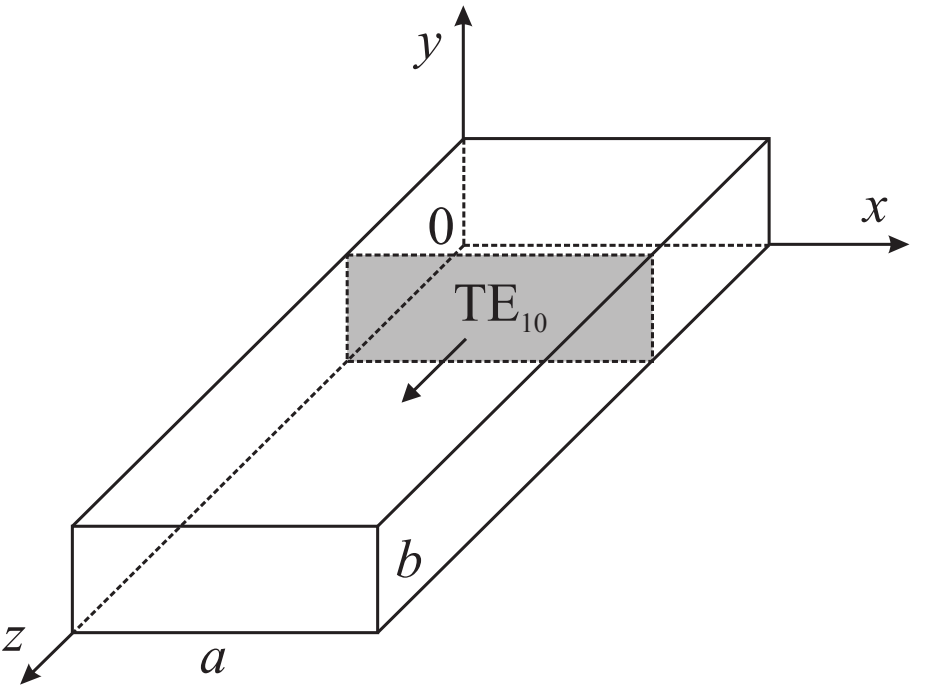


Fig. 1.8. Rectangular waveguide

elements, meaning there is 6 transfers without halo elements and 3 arrays must be transferred with halo elements, resulting in $3/(6+3) = 1/3 = d_h$.

In the complexity model we use the following parameters of GPU2 card (then the occupancy $\varrho = 2/3$): $\beta_h = 3.27$, $W(2/3) = 83.0742 \cdot 10^9/4$ (elements/s), $\gamma = 1.60E-12$, $N = 256^3 \cdot 10 \cdot 18$, $\bar{N} = 256^3 \cdot 10 \cdot 36$. Now we can predict the computation time

$$T = (1 + \beta_h d_h)N/W(\varrho) + \gamma\bar{N} = 0.314 \text{ seconds.} \tag{1.16}$$

The real calculations took 0.177 seconds. So our model predicts correctly the main order of CPU time. The reasons why real calculations are faster than our prediction may be explained by the specificity of an algorithm – compiler optimizes implementation to use the feature (which is supported on GPU2) of simultaneous sending operation in both directions: from global memory to shared and from shared to global. In these calculations the data is transferred in both directions and our model does not take this effect into account, but to get an upper-bound (pessimistic) estimate such assumption works well.

Video cards of compute capability 2 and higher support data caching technology, that acts similarly to the shared memory, but the optimization is done automatically. Can it replace the shared memory? The answer is positive at least in some cases. In Table 1.5 we present results of experiments with different approaches. Also we present results for a modified data splitting, when data is split into 3D blocks, since for halo elements data caching is done automatically. We see that L1 cache 3D approaches give similar results. Also we present the actual computing power in GFlops that was obtained. The results of CPU parallel version on 4 core processor core-i7-860 2.80 GHz are also presented (CPU version used all 4 cores) – its is slower approximately 10 times than GPU version.

Table 1.5. The comparison of three approaches

approaches	times(s)	GFlops	GB/s
shared memory	0.177	34.1	68.2
L1 cache	0.232	26	52
L1 cache 3D	0.179	33.7	67.4
CPU	2.5	2.42	–

We note that until this point we used a standard cross stencil only. But for practical purposes so-called box stencils may be used. We have also implemented the box stencil based calculations with all halo elements being stored in shared memory. In this case there are 27 neighbors for each element. We have done computational experiments with three different patterns of stencils. First, we have used the standard cross stencil, when seven points define the set of neighbors. In this case we have analyzed two scenarios, when only three points of the set are used and when all seven points are used. Second, we applied the full 3D box stencil with 27 neighbors. Note, that this stencil enables also the implementation of algorithms requiring smaller stencils (such as cross stencil). In Table 1.6 the results of experiments are presented, where different stencils are used. The cases of shared memory and L1 cache usage are compared.

Table 1.6. The comparison of times of three approaches

method	3 of 7	7 of 7	27 of 27
L1 cache	0.179	0.266	0.711
shared memory, Cross stencil	0.177	0.186	–
shared memory, Box stencil	0.227	0.226	0.231

As we see from Table 1.6 with all halo elements being stored in shared memory the number of neighbor elements used have no effect. Note that if we compare the usage of box stencil implementation to the native cross stencil implementation it has no significant negative impact on efficiency even when 3 or 7 elements of the box stencil are used. So the box stencil can be used as a good universal alternative to implementation oriented to cross stencil calculations.

1.6. Summary of the First Chapter

In this chapter the CUDA implementation for solving numerical problems described on a regular mesh with a fixed stencil was proposed and investigated. The detailed computations analysis was performed on different GPUs, the model for prediction of computations time was proposed. This model lets to evaluate different GPUs in terms of their efficiency for 3D FDTD calculations. However, some assumptions, that were made to create that model, are not valid for GPU3, so it should be modified to be more general so it could be applied to newer GPUs. GPU version of the solver gave reasonably fast computations times, by doing calculations around 10 times faster than CPU 4 core parallel version.

It is shown that the bottleneck of calculations is usually described by GPU bandwidth parameter. Additionally the possibility of efficient usage of L1 cache memory is demonstrated in computational experiments. As universal alternative Box stencil approach is proposed – it is almost as good as Cross stencil. However it should be noted that its much easier to implement Box stencil even if you need to use Cross stencil only, on the other hand with bigger stencil radius in Box stencil case shared memory limitation can become an issue. Also the possibility of efficient usage of L1 cache memory is demonstrated in computational experiments. However L1 cache efficiency strongly depends on compiler that is sensitive to details of implementation, and cannot guarantee that the calculations will be performed in an optimal way. The real technological problem was solved using this implementation, results were presented in another dissertation by Šlekas (2014).

Lets conclude the technical details of time prediction model. The formula is

$$T = (1 + \beta_h d_h)N/W(\varrho_0) + \gamma\bar{N}, \tag{1.17}$$

where parameters β_h and γ depends on GPU model and are calculated exper-

imentally using proposed benchmarks. d_h, N, $\bar{N}$ depends on computational model and parameters. $W(\varrho_0)$ is, however, tricky to calculate because we need the occupancy of the final implementation ϱ_0, since it is necessary to know registry count to get it (this number depends on compiler, its options and implementation details). However the function $W(\varrho)$ is determined using special benchmark.

1.7. Conclusions of the First Chapter

1. The proposed model for prediction of computations time, which is based on detailed computations analysis, lets to evaluate different GPUs in terms of their efficiency for 3D FDTD calculations. However, the model should be modified to be more general so it could be applied to a newer GPUs.
2. The bottleneck of calculations is usually described by GPU bandwidth parameter. Additionally there is the possibility of efficient usage of L1 cache memory (this demonstrated in computational experiments).

2

The modelling of heat transfer in electrical cables

In this chapter we will discuss how to apply Finite volume method efficiently to problems with complex geometrical domains and values of coefficients of materials may have big jumps on contours of different subdomains. FVM was applied to model heat transferring in electrical cables. An open source software OpenFOAM was used to perform calculations. Results were validated using independent FVM solver implementation written on C++. Due to a problem specificity it was identified some efficiency issues related to geometry and material parameters. Two different strategies of generating spatial meshes for heat transfer simulation in underground electrical cables were discussed in this work. Our proposed mesh generation approach showed much better results than OpenFOAM default one. In order to obtain this result original OpenFOAM code was modified. Heuristic for generating adaptive spatial mesh is proposed. To control mesh singularities when modelling some real life cases the special algorithm to control mesh singularities is proposed and analyzed.

The problem is solved using distributed calculations, scalability analysis is performed.

Parts of this chapter are published in (Bugajev, Čiegis 2012), (Jankevičiūtė, Leonavičienė, Čiegis, Bugajev 2013), (Čiegis, Suboč, Bugajev 2014), (Bugajev, Jankevičiūtė, Suboč, Tumanova 2014), (Čiegis, Starikovičius, Bugajev 2014), (Čiegis, Starikovičius, Bugajev 2014a).

2.1. Introduction Into Second Chapter

This chapter is dedicated to proposal of technological solutions for developing design rules for power transmission lines and cables (Fig. 2.1, (Dongping, Zhang 2009)), which have to meet the latest power transmission network technical and economical requirements. In order to do that it is necessary to de-

Fig. 2.1. Typical high-voltage (110 kV) cables (Dongping, Zhang 2009)

velop specific software solutions. At present, sizes of the power lines are up to 60% bigger than is necessary in terms of transmitted power. However, as the new distributed generating capacities are installed e.g. large wind farms, bio-gas plants or waist-to-energy plants, the infrastructure of power grid must be re-designed or new optimization strategies for the available grid must be developed. Power cables for power distribution applications are still rated according to IEC 287 and IEC 853 standards, which use the Neher and McGrath methods proposed in 1957 (Neher, McGrath 1957). Obviously, these formulas cannot accurately account for the various conditions under which the cables are actually installed and used. They estimate the cable's current-carrying capacity (so-called *ampacity*) with significant margins to stay on the safe side (Makhkamova 2011). The safety margins can be quite large and

result in 50–70% usage of actual resources. A more accurate mathematical modelling is needed to meet the latest technical and economical requirements and to elaborate new, improved, cost-effective design rules and standards. Today there are many applications where analytical and heuristic formulas cannot describe precisely enough the conditions under which the cables are installed. The present standards require that the cable's current-carrying capacity must be reduced according to the worst-case scenario. To be on the safe side this rule is acceptable, but today the cost effective designing of cable installations comes first as the copper price level has reached its maximum value.

When we need to deal with mathematical models for the heat transfer in various media (metals, insulators, soil, water, air) and non-trivial geometries, only the means of parallel computing technologies can allow us to get results in an adequate time. To solve numerically selected models, we develop our numerical solvers using the OpenFOAM package (OpenFOAM 2015).

The knowledge of dynamics (in time) of heat distribution in/around electrical cables is necessary to optimize the usage of electricity transferring infrastructure. It is important to determine: maximal electric current for the cable; optimal cable parameters in certain circumstances; cable life expectancy; other engineering factors. To solve the optimization problem it is necessary to implement an efficient modelling software for heat distribution in cables. Fundamentals of the heat distribution in cables are given in (Incropera, DeWitt, David 1985), but for further readings refer (Ilgevicius 2004; Ilgevicius, Liess 2003; Taler, Duda 2006). Čiegis, Ilgevičius, Liess, Meilūnas, Suboč (2007) and Čiegis, Meilūnas, Jankevičiūtė, Starikovičius (2008) presented efficient parallel numerical algorithms for simulation of temperature distribution in electrical cables for mobile devices and cars and solved inverse problem for fitting the diffusion coefficient of the air-isolation material mixture to the experimental data. Numerical algorithms for parabolic and elliptic problems with discontinuous coefficients have been widely investigated in many papers. The use of standard finite element method (FEM) to solve interface problems is equivalent to arithmetic averaging of discontinuous coefficients. The mixed FEM leads to the harmonic averaging if special quadrature formula are used – see, e.g. works by Falk, Osborn (1994) and Ilgevicius (2004). Conservative finite-difference schemes for approximation of parabolic and elliptic problems were derived by Samarskii (2001) and Tichonov, Samarskii (1961). These schemes are robust and use only general assumptions on the position of the interface. Also such finite difference schemes were proposed, which approximate with the second order of accuracy both – the solution and the normal flux through the interface – see (Il'in 1996; LeVeque, Erratum 1995) for details.

As it was mentioned, current IEC standards consider only standard electrical cables and the safety factors are large, that can result in 50-70% usage of actual resources. The mathematical modelling can reduce cable usage restrictions, by taking into account different circumstances precisely. To reach this goal it is necessary to propose the modeling approach that would allow to create the efficient software. The software need to be as efficient as possible because it is going to be used in variety of optimization problems, some of which can be time consuming and the efficiency is the main factor limiting the quality of optimization results.

Finite Volume Method

In this research the modelling of heat transfer processes is performed using the Finite Volume Method (FVM). The idea of this method is to discretize the modelling area by finite volumes to create a mesh and then apply FVM to construct the system of linear equations (see (LeVeque 2002)). The system of linear equations is solved to obtain the values of temperature. According to (Eymard, Gallouet, Herbin 2003) there are two reasons to chose FVM instead of FEM or FDM:

1. The finite difference method becomes difficult to use when the coefficients involved in the equation are discontinuous (e.g. in the case of heterogeneous media). With the finite volume method, discontinuities of the coefficients will not be any problem if the mesh is chosen such that the discontinuities of the coefficients occur on the boundaries of the control volumes.

2. Comparing to finite difference method from the industrial point of view, the finite volume method is known as a robust for the discretization of conservation laws (by robust, we mean a scheme which behaves well even for particularly difficult equations, such as nonlinear systems of hyperbolic equations and which can easily be extended to more realistic and physical contexts than the classical academic problems).

Open source software OpenFOAM (OpenFOAM 2015) was used as a base for implementation of FVM calculation schemes. OpenFOAM is a free, open source CFD software package. It has an extensive set of standard solvers for popular CFD applications. It also allows us to implement our own models, numerical schemes and algorithms, utilizing the rich set of OpenFOAM capabilities (Weller, Tabor, Jasak, Fureby 1998). The important consequence of

this software development approach is that our numerical solvers can automatically exploit the parallel computing capabilities already available in the OpenFOAM package.

At the same time the independent solver (written in C++) was developed in order to confirm the results of OpenFOAM. All the results presented were equivalent with both solvers. Discontinuities of coefficients make it necessary to discretize the domain by finite volumes, which would fit to the contours of different sub-domains precisely. So it is necessary to generate special meshes that are adapted to the geometry of the domain. The complicated geometry of domains with discontinuous coefficients can cause so-called non-orthogonality of a mesh.

The non-orthogonality of a mesh. Fig. 2.2 illustrates non-orthogonality of a mesh of two finite volumes. Non-orthogonality of the mesh and additional error appear, when the vector connecting the centres (P and N) of neighbouring elements is not collinear to the normal vector $\overrightarrow{n}$, (i.e. $\alpha \neq 0$), then R is not at the center of the edge F. It is a well known problem for non-structured grids, more details can be found, for example, in (Loudyi, Falconer, Lin 2007).

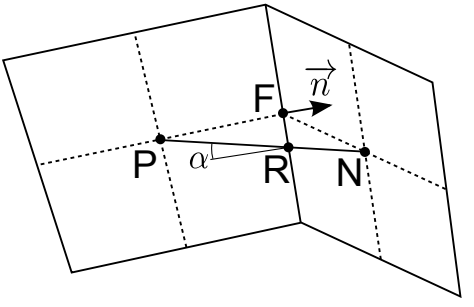


Fig. 2.2. The illustration of non-orthogonality of the mesh

2.2. Mathematical Modelling Problem

The main aim of research is to develop and validate a set of solutions for the heat transfer simulation software in underground electrical cables (see Fig. 2.1 and Fig. 2.3).

There are two possible arrangements of cables in the installation:

1. Single core arrangement.

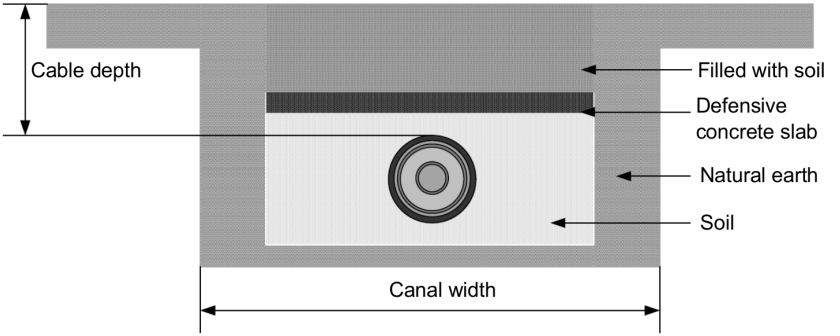


Fig. 2.3. Underground layout in canal

2. Three-core arrangement.

In each case a group of cables can be arranged horizontally or vertically (i.e. Fig. 2.4).

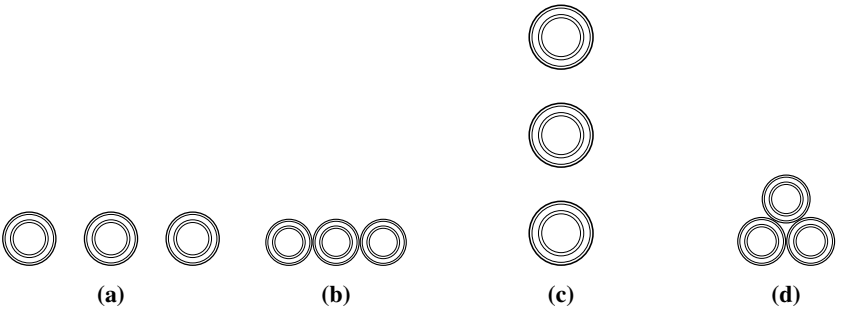


Fig. 2.4. Examples of cables layout topologies: (a) and (b) are three single core cables arranged horizontally; (c) three single core cables arranged vertically; (d) three-core cable

Since the heat transfer mechanism in the underground electrical cables is very complicated, the model should be simplified. In general cable consist of metallic core and slices of different materials around it. For mathematical correctness tests it is enough to take one slice with typical parameters. The length of a cable is much bigger than its diameter (there are no effects along the cable's length), soil (or sand) area is much bigger than the area of cables, so considering the two-dimensional model in soil is sufficient for testing general ideas and solvers. An example of domain description for mathematical modelling of the problem is presented in Fig. 2.5. The metal area (with thermal source) is marked with red, soil – grey, isolation – blue color. It has to be noted that the heat conductivity of isolation is much smaller than heat con-

ductivity of metal and area of cables in the whole domain is relatively small ($h_{2,3} \gg d$).

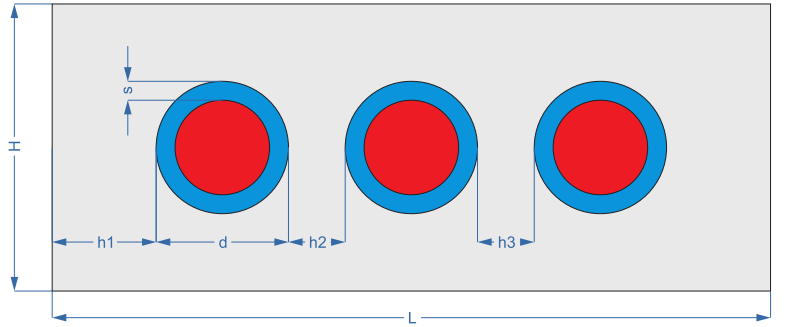


Fig. 2.5. An example of geometry: three cables in the soil

The object that must be implemented in modelling software should be described by appropriate mathematical model. So first, the mathematical model of the problem should be formulated. The non-stationary heat conduction problem is given by the parabolic differential equation (Čiegis, Ilgevičius, Liess, Meilūnas, Suboč 2007):

$$\begin{cases} c\rho\dfrac{\partial T}{\partial t} = \nabla\cdot(\lambda\nabla T) + q, & t \in [0, t_{max}], x \in \Omega\backslash\partial\Omega_D, \\ T(x,0) = T_b, & \text{when } x \in \Omega, \\ T(x,t) = T_b, & \text{when } x \in \partial\Omega, \\ [T] = [\lambda\nabla T] = 0 & \text{when } x \in \partial\Omega_D, \end{cases} \tag{2.1}$$

here $x = (x_1, x_2)$, $T(x,t)$ is temperature, $\lambda(x) > 0$ is heat conductivity coefficient, $q(x,t,T)$ is the source function, $\partial\Omega$ is the contour of domain Ω, $\rho(x) > 0$ defines mass density, $c(x) > 0$ is specific heat capacity, T_b, t_{max} are given constants. Operator $\nabla\cdot(\lambda\nabla T) = \sum\limits_{j=1}^{2} \frac{\partial}{\partial x_j}\left(\lambda\frac{\partial T}{\partial x_j}\right)$ is the diffusion operator. The solution and flux continuity conditions are satisfied on boundaries of domains with different diffusion coefficients $\partial\Omega_D$ The values of parameters used in computational tests are the following:

$$\begin{aligned}
&\rho_{metal} = 8700, \quad c_{metal} = 385, \quad \lambda_{metal} = 400,\\
&\rho_{isolation} = 1380, \; c_{isolation} = 2000, \; \lambda_{isolation} = 0.28,\\
&\rho_{soil} = 1600, \quad c_{soil} = 890, \quad \lambda_{soil} = 1.
\end{aligned}$$

In case of three cables the domain is described by the following parameters (geometrical values are given in [m] from Fig. 2.5):

$$\begin{aligned}
&d = 0.0348, s = 0.0076, H = 0.2, L = 0.4;\\
&h1 = 0.0826, h2 = h3 = 0.0652
\end{aligned}$$

For standalone cable from (Fig. 2.6) we use the following parameters:

$$W = H = 0.2, R_1 = 0.0174, R_2 = 0.025.$$

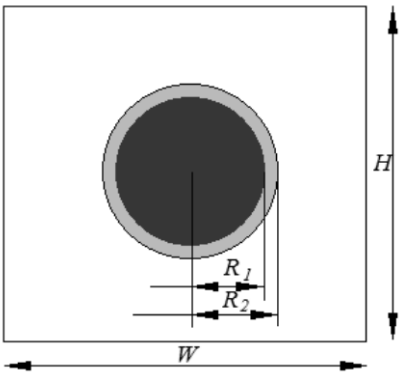


Fig. 2.6. Geometry of simplified cable

The FVM method was used to solve this problem numerically. The main goal for a software implementation is the efficiency of calculations. Next we will formulate the main remarks about the formulated model, that make it difficult to create the efficient modelling software:

1. The coefficients λ, c, ρ are discontinuous, therefore it is necessary to adapt the mesh to discontinuity contours that forces to use non-orthogonal mesh. The usage of FVM on a non-orthogonal mesh can create the non-orthogonality error that makes the implementation less efficient.

2. The values of coefficients may differ thousands times.
3. The size of cables can be much smaller than the whole modelling area.
4. The cables can touch each other leading to mesh singularity problems.
5. The source function $q(t, T)$ can change in time very fast at some intervals and can be constant for long periods of time later.

Here we present our approaches to solve these problems, that will be described in details later:

1. The usage of triangulation by acute triangles and calculation of Voronoi centers solves the flux non-orthogonality problem.
2. Harmonic mean formulas for diffusion coefficients improve the efficiency, in the case of big jumps of coefficients. When OpenFOAM is used to create a solver this approximation is implemented by selecting appropriate option from a list of possible approximations. Tests showed that this option is critical, it improves the order of efficiency, so it is used in all calculations in this work.
3. Adaptive meshes are used. The mesh is coarse in areas, that are far from cables, such a strategy helps the solver to avoid unnecessary calculations.
4. To automatically avoid the specific mesh singularities a special algorithm is constructed.
5. Adaptive time steps should be used, such a strategy avoids detailed time calculations where it is not needed. But this approach is out of the scope of this research.

2.3. Modelling Approach

In this section approaches for solving calculations efficiency problems will be presented.

2.3.1. Finite Volume Method Software

There many FVM software packages exist. The essential criteria that the package should meet is to be open-source, because first of all it is needed to be open for modifications if the improvements needed to be performed are identified. Here we present some of these packages: OpenFOAM (OpenFOAM

2015), SU2 (Palacios, Economon 2012), OpenFVM (OpenFVM 2015), Clawpack (LeVeque 1994), Gerris (Popinet 2001), Code_ Saturne (Electricity of France 1997). However, some of them are oriented to model Computational fluid dynamics (CFD), they are: Code_ Saturne, OpenFVM, Gerris. SU2 and Clawpack are oriented to use predefined solver, and it may be tricky to implement a custom solver for custom system of differential equations. However, OpenFOAM is implemented as a toolbox for the development of customized numerical solvers. It is done in a such way that user needs to write the differential equations himself – this ensures the comfortable degree of freedom not only to perform the research we need in this dissertation but also make it easier to extend the results to be applied to the wider range of problems, e.g. Multiphysics models with very specific equations. Here we present the disadvantages of OpenFOAM comparing to any other mentioned packages:

- It has no integrated graphical user interface. However, it has powerful visual postprocessing tool paraFoam.
- It may be hard to learn to use OpenFOAM, because the official guides often do not provide sufficient details.
- This software has many options to configure the solving process, making it universal but sometimes complicated to use in optimal way – the professional experience is needed and sometimes additional investigation must be performed.

However, mentioned drawbacks have no negative effect on the quality of results in this dissertation, so OpenFOAM is chosen as a base for solver implementation.

2.3.2. The Benchmark to Control the Efficiency

The most of formulated above efficiency problems do not depend on time t in model (2.1). The exception is adaptive modelling of dynamics in time, when the adaptive time steps can be used. So for testing purposes we consider the simplified stationary model ($t_{max} \to \infty, q(x,t) = q(x)$) of single cable with known solution. We note that efficiency is directly related to numerical error, since smaller error lets to use less finite volumes to compute the same result – so improving the numerical method accuracy is equal to improving the efficiency. To control the time independent accuracy of numerical approximation

the simplified model is used:

$$\begin{cases} -\nabla \cdot (\lambda \nabla T) = q, & x \in \Omega \backslash \partial\Omega_D, \\ T(x_1, x_2) = u(x_1, x_2), & \text{when } (x_1, x_2) \in \partial\Omega, \\ [T] = [\lambda \nabla T] = 0 & \text{when } (x_1, x_2) \in \partial\partial\Omega_D, \end{cases} \tag{2.2}$$

Take one cable of core radius R_1 and one insulation layer (Fig. 2.6), $t_{max} \to \infty, \quad q(x,t) = q(x)$, when the exact solution is known.

Then the coefficients are defined as:

$$q(x_1, x_2) = \begin{cases} I, & \text{when } 0 \le r \le R_1, \\ 0, & \text{when } r > R_1, \end{cases} \tag{2.3}$$

$$\lambda(x_1, x_2) = \begin{cases} \lambda_1, & \text{when } 0 \le r \le R_1, \\ \lambda_2, & \text{when } R_1 < r \le R_2, \\ \lambda_3, & \text{when } r > R_2, \end{cases}$$

where heat conductivity coefficients $\lambda_1, \lambda_2, \lambda_3$ are defined for metal, insulation and soil, correspondingly. r is polar coordinate $r = \sqrt{x_1^2 + x_2^2}$.

The exact solution of stationary problem (2.2) is given by the following functions:

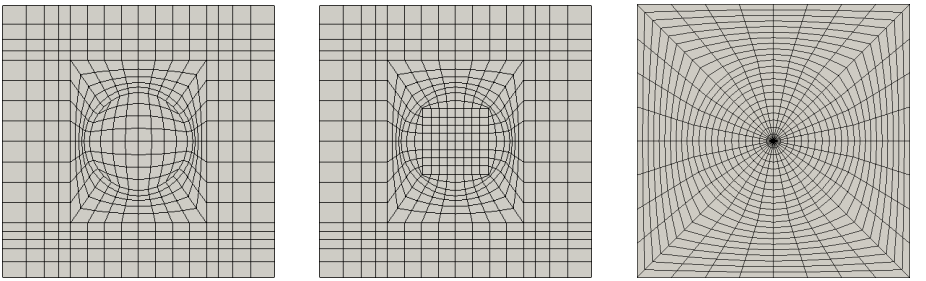
$$u(x_1, x_2) = \begin{cases} \dfrac{1}{\lambda_1}\big(C_1 - Ir^2/4\big), & \text{when } 0 \le r \le R_1, \\ \dfrac{C_2}{\lambda_2}\big(\ln r - \ln R_1\big) + C_3, & \text{when } R_1 < r \le R_2, \\ \dfrac{C_2}{\lambda_3}\big(\ln r - \ln R_2\big) + C_5, & \text{when } r > R_2, \end{cases}$$

where the constants are:

$$C_2 = -\frac{R_1^2}{2} I, \quad C_1 = \frac{R_1^2}{4} I + \lambda_1 \left[T_b - C_2 \left(\frac{1}{\lambda_2} \ln \frac{R_2}{R_1} + \frac{1}{\lambda_3} \ln \frac{R_3}{R_2} \right) \right],$$

$$C_3 = \frac{1}{\lambda_1} \left(C_1 - \frac{R_1^2}{4} I \right), \quad C_5 = \frac{C_2}{\lambda_2} \ln \frac{R_2}{R_1} + C_3,$$

here R_3 is the distance from the center of cable at which the temperature $u(x_1, x_2)$ is equal to T_b, $u(x_1, x_2) = T_b, \text{when} x_1^2 + x_2^2 = r^2$. We note, that



(a) (b) (c)

Fig. 2.7. Examples of structural meshes generated by built-in OpenFOAM mesh generation tool

in most cases the maximizing efficiency of the solver means efficient way to minimize the error, so this benchmark will be used to control the error for software tests.

2.3.3. Non-orthogonality Error Problem Solving

There exist several strategies to solve the non-orthogonality problem, we observe some of them:

- Correction of the approximation by using iterative methods (supported by OpenFOAM directly).
- To use dual meshes from the basic Delaunay triangulation.
- To triangulate domain with acute triangles and take Voronoi points as cell centres.

Since the contours of dual mesh finite volumes are not aligned to the contours of domains with different coefficients, the dual mesh construction is more problematic for solution of this problem. That is why only the first and the last strategies are considered.

Non-orthogonality iterative corrections strategy. The OpenFOAM build-in iterative non-orthogonality corrections approach was tested by solving the benchmark problem. Firstly, the rectangular mesh using OpenFOAM built-in tools was constructed (Fig. 2.7). Then the OpenFOAM built-in non-orthogonality correction algorithms were tested and the convergence of solution was analysed. The results of this analysis are presented in Table 2.1.

In this table:

- N – the number of finite volumes,

Table 2.1. Convergence of FVM algorithm with iterative corrections for non-orthogonal meshes

N	E_C	p
704	0.2729	-
2816	0.1432	0.93
11264	0.0718	1.00
45056	0.0353	1.02
180224	0.0174	1.03

- E_C – defines the error in C norm, the maximal difference between exact and numerical solutions for all mesh volumes,
- p – the experimental convergence order, i.e. $E_C = O(h^p)$,
- h – the average size of triangles. The size of triangle is equal to the biggest length of triangle edges.

Tests showed that $h = O(N^{1/2})$, so we assume, that $E_C = AN^{-p/2}$, where A is a constant, which is not important. The results presented in Table 2.1 show that the convergence rate is lower than theoretical rate of FVM ($p = 2$) and non-orthogonality error is dominant. Error distribution is presented in Fig. 2.8. As it can be seen, the error distribution is not radially symmetrical (as the solution is). Just for comparison the error distribution with orthogonal rectangular mesh (without non-orthogonality error) within circular domain is also presented (Fig. 2.9). This mesh does not fit well the analysed case, since boundaries of the domain are circular, but the problem needs to be calculated in a rectangular domain. Fig. 2.9 represents the expected behavior of error distribution for the case when non-orthogonality error is not dominant.

Discretization with acute triangles. Another way to solve non-orthogonality error problem is to use triangular mesh and to take Voronoi points as finite volumes centres that form an unstructured orthogonal mesh (Fig. 2.10).

However, this method puts a strict requirement on the mesh: on the mesh all angles of all triangles must be less than 90 degrees. In other words – triangles must be acute. To obtain such meshes the aCute library (University of Florida 2009) was used. The corresponding examples of triangulations are presented in Fig. 2.11 and Fig. 2.12a.

It should be noted that due to this requirement (for triangles to be acute) in some cases the mesh becomes much more dense in some domain places, however, for most real world applications such mesh specificity is do not change

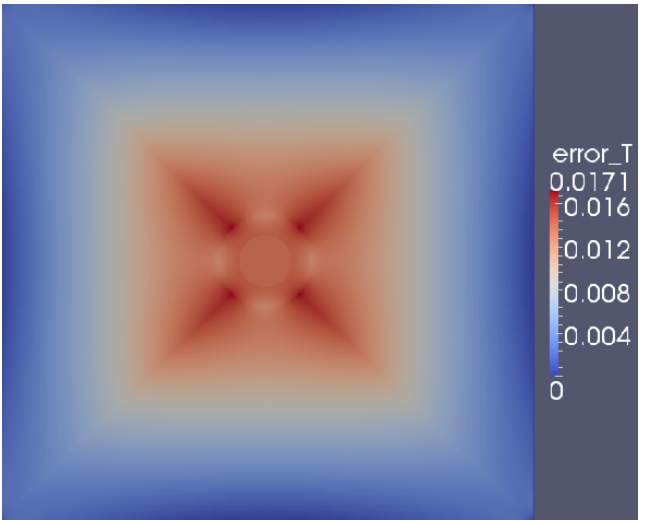


Fig. 2.8. The error distribution with $N = 180224$ using the mesh topology from Fig. 2.7c

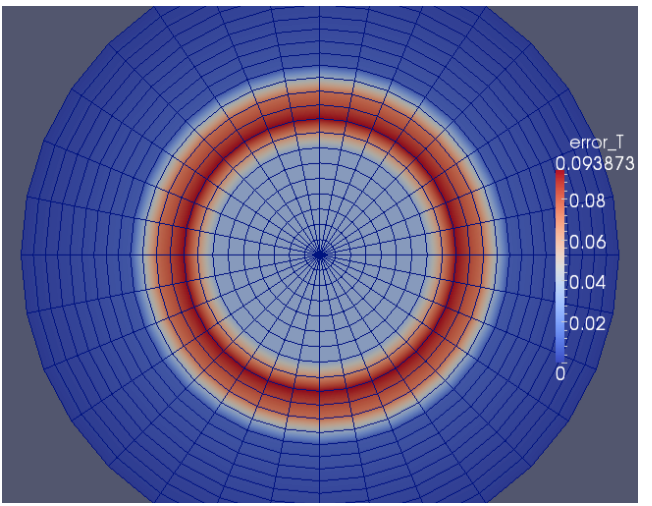


Fig. 2.9. The error distribution using the radially symmetrical mesh topology ($N = 704$)

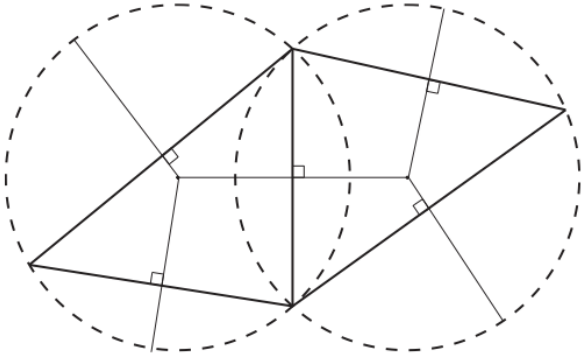


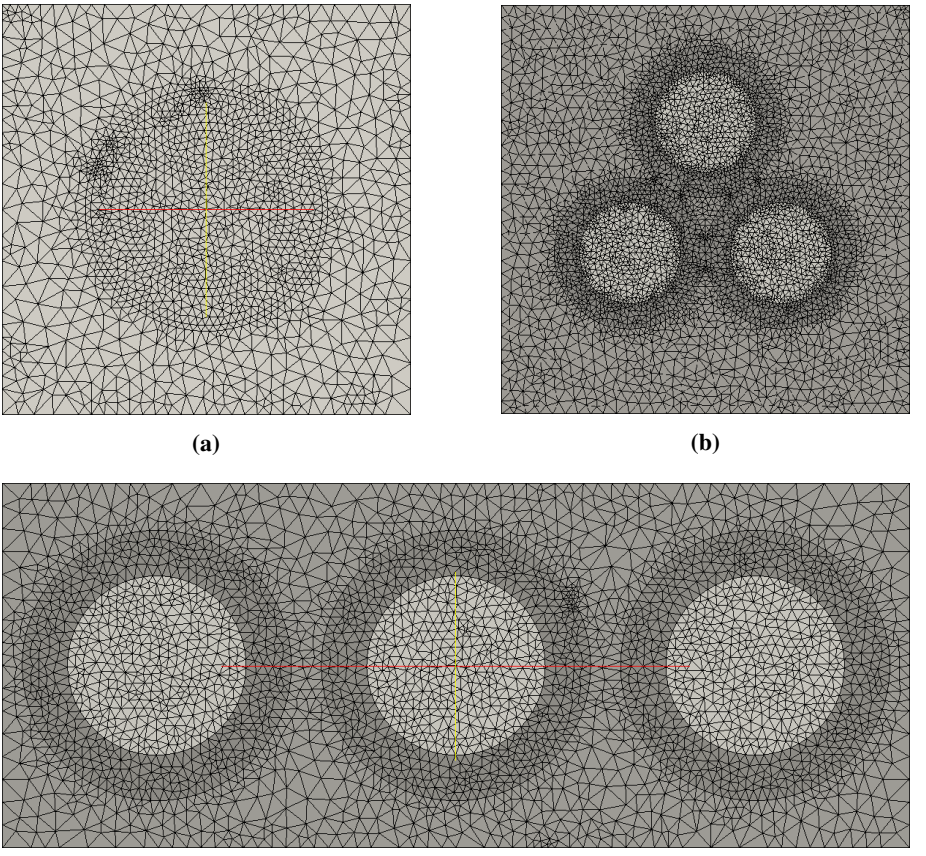
Fig. 2.10. Mesh orthogonality with acute triangles

the order of the total number of triangles. So such mesh has no significant negative effect on efficiency of computations.

The special mesh generator Mesher was developed to generate aCute meshes from xml geometry configuration files and to convert these meshes into OpenFOAM polymesh format. OpenFoam does not support the usage of acute triangles directly and even does not provide any possibility to configure it for acute triangles. However OpenFOAM is an open source software, so in order to implement acute triangular orthogonal mesh based solver we have modified OpenFOAM. The modified version of OpenFOAM supports acute triangles 2D meshes and it is constructed specially for needs of this work.

The convergence analysis is presented in Table 2.2. First let's make additional notes about experimental convergence rate. Refinement cannot be done recursively and uniformly, since the mesh must fit coefficients discontinuity contours. Therefore it is not possible to guarantee that in the whole mesh all triangles become smaller in equal proportions. The only way is to set needed parameters for aCute software, so that the obtained refinement is pseudo-uniform. The results presented in Table 2.2 show that the convergence rate is close to 2.

The visual comparison of results of Table 2.2 and Table 2.1 are presented in Fig. 2.13. We note, that both vertical and horizontal axis are in logarithmic scale, so the differently directed lines represent a big difference in efficiency of different strategies.



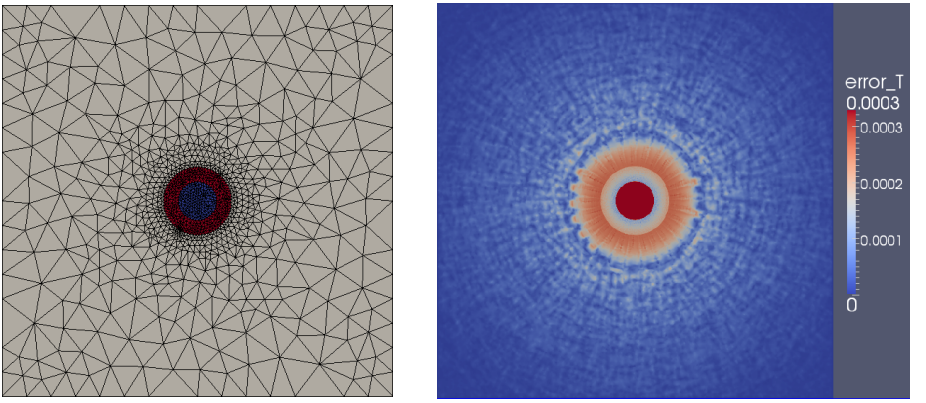
(a) **(b)**

(c)

Fig. 2.11. Meshes generated with aCute software for (a) stand alone cable, (b) compact triangular placement of three cables, (c) linear placement of three cables

2.3.4. Adaptive Meshes

Introduction. The uniformly refined mesh is a good mesh when physical effects are uniformly distributed on the whole modelling area. However, when important effects are localized in a small part of the whole modelling area this part must be refined accordingly to model effects in this small area. Also note, that it is not needed to refine the rest part of area with more finite volumes since there are no significant effects in it. So the usage of uniform mesh in situation when local effects are simulated can drastically increase the number of finite volumes. It can also be described by information compressing analogy:



(a) **(b)**

Fig. 2.12. Calculations with $N = 435472$ and the mesh topology: (a) mesh topology, (b) error of discrete solution

Table 2.2. Convergence for non-orthogonal acute meshes

N	E_C	p
669	0.20077	-
2230	0.04830	2.37
7061	0.01495	2.03
27639	0.00500	1.61
110069	0.00132	1.93
435472	0.00033	2.01

the finite volumes in the big part of mesh have small amount of information (description of effects) so it can be compressed by taking bigger finite volumes at the cost of approximation error that have no significant effect on the global error of the whole solution.

In such cases the usage of adaptive mesh is critical, comparing to uniform mesh it can easily decrease the number of finite volumes 100 times or even more, also decreasing the computation times by the same order. As it was mentioned before, in Fig. 2.5, $h_{2,3} \gg d$, $L, H \gg d$, in other words the cables are relatively small and the heat sources are defined in cables only, so the cables and the areas near them have the biggest gradient of solution, and physically the fastest (in time) changes of solution are also localized here. That is why this section is dedicated to adaptive meshes – this technique is necessary to make modelling process efficient.

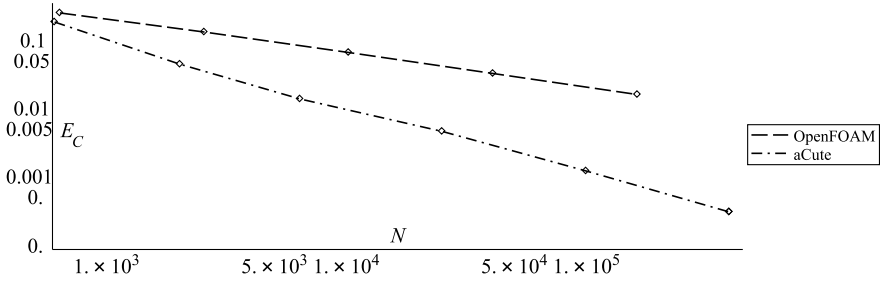


Fig. 2.13. Comparison of non-orthogonal acute meshes strategy and OpenFOAM FVM algorithm with iterative corrections for non-orthogonal meshes

Reducing the size of problem. As it was already mentioned domain discretization with uniform mesh can drastically increase the size of computations. This problem is a part of a general idea of choosing a method that simplifies unneeded calculations and the adaptive mesh is only one of strategies to reach this aim.

There exist many ways how to derive a simpler numerical model from a complex one. Derivation of simple models in general means derivation of models which comprise less number of equations or variables and which are numerically fast for computations or simulations. Such simple models can be derived based on physical assumptions or based on data collected from simulations or experiments. Hence there are two main approaches for the derivation of simpler models (Astrid 2004):

- Physical-insight based approach. Using physical insight, an initially complex model can be transformed into a simpler one by considering its physical phenomena.
- Black-box modeling approach. With the advent of system identification techniques, such as subspace identification and neural networks, empirical models can be derived from the input output data.

Model order reduction (MOR) techniques try to reduce the computational complexity and computational time of large scale dynamical systems. The idea is to approximate the given large problem by a problem of much lower dimension that can produce nearly the same response characteristics. Reduced-order modeling is a powerful and ubiquitous tool and it have been given significant attention in recent years. Research in the area of reduced-order modeling has followed two approaches:

- non projection based methods,

- projection based methods.

The vast majority of research is concerned with projection based methods, such as the balanced truncation and POD. There are many papers, where such methods are analyzed and applied for problems in heat transfer (Fic, Białecki, Kassab 1993), fluid mechanics, lithium-ion battery (Cai, White 2009), reactor system (Marquez, Oviedo, Odloak 2013). The POD method is described in details e.g in (Berkooz, Holmes, Lumley 1993; Sun, Luo, Zhou 2010).

In the article "Reduced order models based on pod method for schrödinger equations" we also show how to develop reduced-order models (ROM) using the proper orthogonal decomposition (POD) for one dimensional linear and nonlinear Schrödinger equations. The study of the accuracy and robustness of the ROM approximations were performed. The sensitivity of generated optimal basis functions on various parameters of the algorithms is discussed. Errors between POD approximate solutions and exact problem solutions are calculated. Results of numerical experiments are presented. It is shown that the reduced order model can be implemented very efficiently by using the precomputed coefficients for the nonlinear interaction term.

However, we solve the problem (2.1) using OpenFOAM that has integrated standard FVM, so the easiest way to reduce the size of numerical problem is to implement the adaptive mesh approach. That is why the rest of this section is dedicated to development of adaptive mesh.

Example of singularly perturbed problem. To develop adaptive mesh technology first we consider a simple best representative one dimensional example of diffusion equation

$$\begin{cases} -\varepsilon^2 u'' + q(x)u = f(x), \quad x \in (0,1), \\ u(0) = 0, \quad u(1) = 0, \end{cases} \tag{2.4}$$

where $f = f(x)$, $q(x) \geq a > 0$ are given sufficiently smooth functions, $0 < \varepsilon \ll 1$. It is well-known that solutions of this problem can have steep exponential boundary layers, therefore a numerical approximation of singularly perturbed problems is a challenging task. A good review on the state of art in this field is given by Kadalbajoo, Gupta (2010); Stynes, Roos, Tobiska (1996).

Starting from the paper of Bakhvalov (1969), efficient numerical algorithms were constructed using special adaptive meshes. These meshes are exponentially stretched within the boundary layers. Shishkin (1986) proposed a simpler type of adaptive meshes, these meshed can be constructed a priori

as a function of the singular parameter ε and are piecewise-uniform. It was proved that such meshes can be used for wide classes of singularly perturbed problems. It is important to note, that if we compare Shishkin and Bakhvalov meshes, Shishkin meshes have almost the same theoretical order of convergence – $O(N)$ comparing to $O(\ln NN)$, where N is the number of mesh nodes. So Shishkin meshes are an efficient and in many cases sufficient approach despite the fact that the construction of these meshes is very simple.

Roos and Linß have proposed simple sufficient conditions to prove a uniform convergence on layer-adapted meshes (Roos, Linß 1999). They have applied these conditions not only for the Bakhvalov and Shishkin meshes, but also have analysed discrete schemes proposed by Boglaev (1984); Vulanovic (2001, 1986). The Bakhvalov and Shishkin meshes were compared in (Vulanovic 2001), where a quasi-linear singularly perturbed boundary value problem was considered and the Shishkin grid was generalized and improved.

The other important trend of theoretical analysis deals with a posteriori error estimates for the numerical solutions of singularly perturbed elliptic equations. Such computable estimates for singularly perturbed or convection-dominated problems where investigated, e.g. in (Angermann 1995; Apel, Lube 1998; Mackenzie 1999; Qiu, Sloan 1999). These estimates can be used to construct non-uniform adaptive meshes by using the equidistribution of a computed approximation of the error monitoring function. The benchmark adaptive meshes can be constructed by the equidistribution of monitoring functions that are based on the exact solution (see, (Qiu, Sloan 1999) for convection dominated test problems).

For convection-diffusion equations different a posteriori error estimators were considered by John (2000), including gradient indicator, residual based error estimators for different norms and two error estimators which are defined by solution of local Neumann problems. The accuracy of numerical approximations was investigated on adaptive meshes. The obtained results show that none of the considered error estimators work robustly in all tests. In (Beckett, Mackenzie 2001), the finite difference approximation of a model singularly perturbed reaction-diffusion boundary value problem (2.4) is investigated. The meshes are based on the equidistribution of a positive monitoring function that takes into account a power of the second derivative of the solution.

Reliable a posteriori estimators for the error in H^1 norm for a 2D singularly perturbed reaction-diffusion model were constructed and investigated in (Kunert 2005). Vulanovic (2007) analyzed the relationship between layer-resolving transformations and mesh generating functions for numerical solution of singularly perturbed boundary value problems. Linß (2007) considered

non-monotone FEM discretization for singularly perturbed problem (2.4). He proved a priori and a posteriori error bounds in the maximum norm.

Non-local boundary conditions are also considered for singularly perturbed diffusion-reaction problems. Exponentially fitted schemes were used to approximate the problem with three-point nonlocal condition by Amiraliyev, Cakir (2002). For integral nonlocal conditions such schemes were investigated by Cakir, Amiraliyev (2005); Čiegis (1988).

We investigate the optimality of the Bakhvalov and Shishkin meshes for singularly perturbed diffusion-reaction problems. We compare these meshes with the adaptive meshes constructed by using general duality-based a priori error estimators in the L_2 and energy norms (Bangerth, Rannacher 2003; Eriksson, Estep, Hansbo, Johnson 1995). A standard technique of the equidistribution of a computed error monitoring function is applied in order to construct optimal meshes for different norms. It is well-known that only the uniform norm enables us to resolve boundary layers for general meshes. The usage of integral L_2 and energy norms is justified for adaptive meshes, which can be seen as special layer-resolving transformations.

For one-dimensional problem the mesh is defined by the list of nodes $x_j, j = 0 \dots N$ Let $W_h : 0 = x_0 < x_1 < \dots < x_{N-1} < x_N = 1$ which is a partition of interval $[0, 1]$.

Simplify (2.4) by taking $q(x) = \rho^2, f(x) = \bar{f}$, where $\rho, \bar{f}$ are constants:

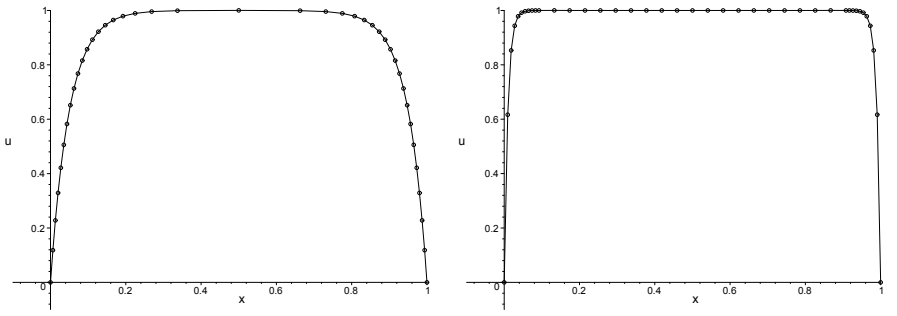
$$\begin{cases} -\varepsilon^2 u'' + \rho^2 u = \bar{f}, & \text{in } x \in (0,1), \\ u(0) = 0, \quad u(1) = 0. \end{cases} \tag{2.5}$$

In appendix A it is shown that both Bakhvalov and aposteriori estimates based meshes gave x_j which is defined by:

$$x_j = -\frac{\sigma\varepsilon}{\rho} \ln\left(1 - \frac{2j}{N}\left(1 - e^{-\frac{\rho}{2\sigma\varepsilon}}\right)\right), \quad j = 1, \dots, N/2. \tag{2.6}$$

Since the mesh is symmetric with respect to point $x = 0.5$, then $x_j = x_{N-j}$ for $j = N/2+1, \dots, N$. The Bakhvalov mesh is exponentially fitted at boundary layers. In Fig. 2.14*a* an example of the Bakhvalov mesh is shown for $f = q = k = p = 1$, $\varepsilon = 0.05$, discretization nodes x_j are marked by dots, $N = 40$. $\sigma = \dfrac{5}{2}$ is proven to be close to optimal when the error is optimized in L_2 norm.

A posteriori estimate based meshes versus Shishkin meshes. A posteriori estimates based mesh also gives Bakhvalov mesh and it is believed to be very



(a) **(b)**

Fig. 2.14. The solution of problem (2.5) and apriori adaptive meshes: a) the Bakhvalov mesh for $N = 40$, $\varepsilon = 0.05$, $\sigma = \frac{5}{2}$, $k = q = f = p = 1$, b) the Shishkin mesh for $N = 40$, $\varepsilon = 0.01$, $\sigma = \frac{5}{2}$, $k = q = f = p = 1$

close to an optimal mesh. However, there is another simple method to construct a mesh when the boundary value problem (2.4) is being solved – a Shishkin mesh. The idea of Shishkin mesh is to separate the whole domain into 2 types of subdomains: regular and singular. In singular domain part the mesh is much more dense than in regular one (example in Fig. 2.14b), but in all subdomains meshes are uniform. For problem (2.5) the Shishkin mesh is computed in two steps: first we define the thickness of boundary layers

$$\lambda = \min\left(\frac{1}{4}, \frac{\sigma\varepsilon}{\rho}\ln N\right).$$

Next the interval $[0, 1]$ is partitioned into three parts: $[0, \lambda), (\lambda, 1-\lambda), (1-\lambda, 1]$. Intervals $[0, \lambda)$ and $(1 - \lambda, 1]$ are uniformly divided into $\frac{N}{4}$ parts, the interval $(\lambda, 1-\lambda)$ is uniformly divided into $\frac{N}{2}$ parts. An example of the Shishkin mesh is presented in Fig. 2.14*b*.

In Appendix A.3. it is shown that the simple Shishkin mesh actually solves the singularity (when the effect is concentrated in indefinitely small area) problem and the error is considerably small. So even for problem (2.4), where the local area of the dominant effects can be as small as it is needed, it is questionable to use a better (in terms of error) mesh than Shishkin mesh. For the case of equation (2.1) the localization of effects is not indefinitely small – the difference of domain and cables sizes are always of the same order. So from this point we assume that a simple heuristic is enough to handle the adaptive mesh.

Automatic adaptivity. The first heuristic is the mesh adaptivity that is obtained using aCute software automatically. It is important to analyze the size function $h(r)$ that shows the dependence of sizes of triangles on a distance r from the domain where much smaller sizes $h(0)$ of the mesh are used. The important characteristic is the speed of adaptivity $h' = \frac{d\,h(r)}{dr}$, it characterizes the strength of adaptivity. Note that for this test the geometrical properties of a mesh are analyzed and the analysis is not directly connected to numerical scheme used to approximate equations, so to compute the distance between triangles the geometrical triangle center are used. To determine $h(r)$ the simple test is done:

1. The whole domain is separated into 2 subdomains with different parameters of sizes of triangles h_{max}.
2. The statistics for the mesh parameters are collected $[r_i, H_i], i = 1, \dots n$ (Fig. 2.17), here n is the number of analyzed triangles, H_i is the size of i-th triangle and r_i is corresponding r.
3. Empirically it is determined the dependency of the sizes H_i on r_i. The determined dependence is approximated by function $h(r)$ which characterizes the adaptive mesh.

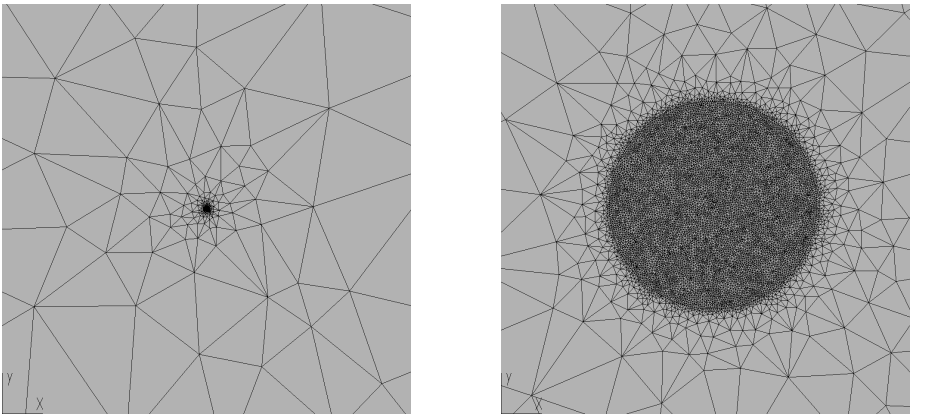
We separate the whole domain into following two subdomains are used for computational tests:

- The circle domain with radius R and parameter $h_{max} = h(0) = 1$, meaning the mesh here is dense.
- The ring that has much bigger outer radius $R_o \gg R$ and parameter h than is needed to measure $h(r), r \leq r_{max} = R_o - R$, the analysis is performed in interval $r \in [0, r_{max}]$. Take for example $R = 100$, $R_o = 20000, h_{max} = 8000$, see Fig. 2.15.

The sizes of triangles are presented in Fig. 2.16, where r is a distance between triangle and subdomain with $h = h(0)$.

$$h(r) = 1.80 + 0.579r. \tag{2.7}$$

From Fig. 2.16 and Fig. 2.15 it is clearly seen that the most triangles are distributed near $r = 0$. Note that there is no need to perfectly fit h to data and prove its optimality but it is needed to predict the bounds of speed of mesh adaptivity. Assume that $h(r)$ is linear function $h(r) = ar + b$, this assumtion



(a) **(b)**

Fig. 2.15. The mesh for adaptivity test (a) and the zoomed version (b) with subdomain with $h = h(0)$.

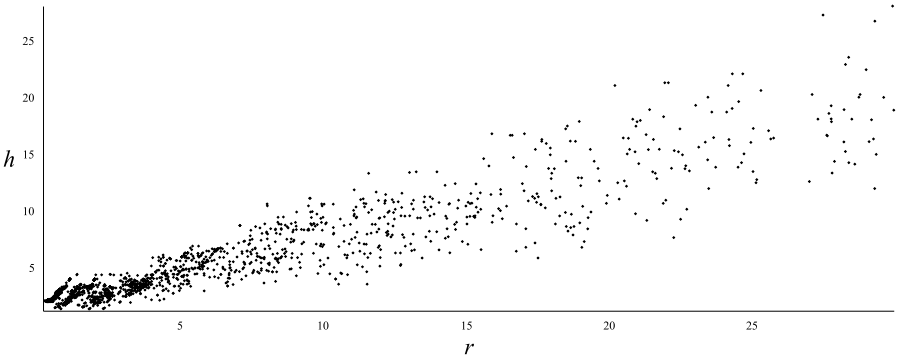


Fig. 2.16. The sizes of triangles

is natural since that means that the speed of adaptivity $h' = a$ does not depend on size h.

Two approaches are considered:

1. Apply the regression analysis to data $[r_i, H_i], i = 1, \dots n$ and by the least squares method find $h(r)$.

2. Separate the interval $r \in [0, r_{max}]$ into m subintervals

$$I_i = \left[(i-1)\frac{r_{max}}{m}, i\frac{r_{max}}{m}\right], i = 1, \dots, m,$$

calculate average values of results in these subintervals $\left[\bar{r_i}, \bar{H_i}\right] =$

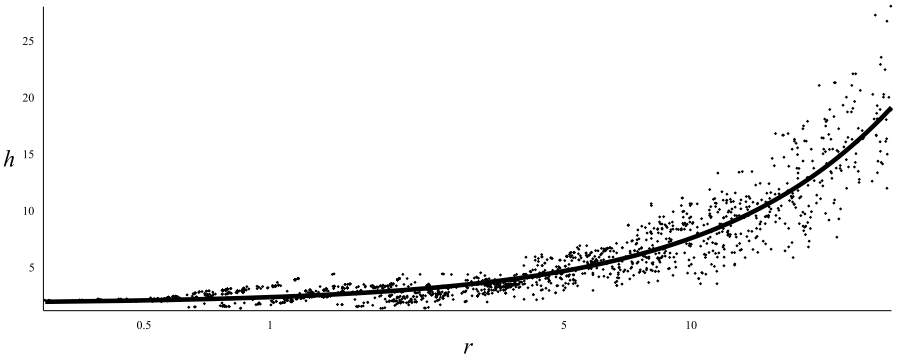


Fig. 2.17. $h(r)$ and data for the first approach

$\left[\sum_{\forall j:r_j\in I_i}\bar{r_j}, \sum_{\forall j:r_j\in I_i}\bar{H_j}\right]$, apply the regression analysis to the average values data.

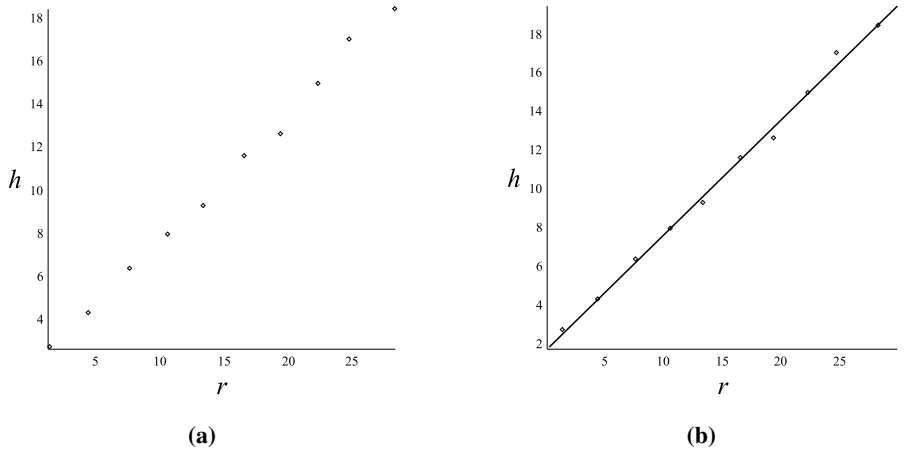
Note that the overwhelmingly large group of triangles is distributed near $h(0)$. The least squires method solves the minimization problem with all triangles having the same weight. However for purposes of this work the triangles near $h(0)$ are just as important as triangles near $h(R_o)$. A classical regression analysis will make size data near $h(0)$ more important. That is why the second approach is presented. In the second approach we split the interval into subintervals so that each interval would have the same weight. This adds additional information lose at averaging step before regression but for linear function it is not critical and tests showed that the number of subintervals $m = 10$ is enough and changing it has no significant effect on the results.

For data on Fig. 2.16 the first approach gave (Fig. 2.17) the second approach gave (Fig. 2.18b)

$$h(r) = 1.63 + 0.594r. \tag{2.8}$$

So both approaches gave similar results. But the second one will be used for future analysis. The speed of adaptivity is $h' = 0.594$ and it means that if the mesh has a ring of triangles with sizes h_0 then the triangles that are farther by distance $h_0/h' = h_0/0.594 \approx 1.68h_0$ have sizes $2h_0$, i.e. are twice larger (see it on Fig. 2.15). The speed of adaptivity is fast enough to solve the most practical problems.

As it was mentioned, the speed of adaptivity $h' = a$ does not depend on size h. Now lets investigate the question if the change of R will have some



(a) **(b)**

Fig. 2.18. The second approach: the average values (a) and the approximation line (b)

effect on results. but will the change of R have effect on the results? We have analyzed the dependence of speed on the inner ring of adaptive zone $h'(R)$. From Table 2.3 it is seen that $h'(R)$ is not a constant. The full analysis of behavior of $h'(R)$ is out of scope of this research, however we note, that making R bigger than 100 have no significant effect on h' and its value never reaches 0.5, so we define the lower bound $0.5 \leq h'$.

Table 2.3. Speed h' for different inner radiuses R

R	h'
100	0.594
50	0.677
25	0.715

The conclusion $h'(R) \neq const$ means that the proposed heuristic do not guarantee the uniform mesh refinement if the h parameter for different subdomains is changed proportionally. To perform convergence analysis a uniform refinement of adaptive mesh is essential.

Controlled adaptivity. In order to control the function $h(r)$ (i.e. regularize its behavior), it can be defined with smaller h' than the software is capable to achieve. The first reason to include this approach is that mesh can change too fast from fine grid to coarse grid making adaptivity too aggressive that can raise additional approximation errors. The second reason is that with properly

defined $h(r)$ the uniform refinement of adaptive mesh can be achieved making possible to perform the correct experimental convergence analysis to evaluate the efficiency of our solver. We propose the following heuristic

$$h(r) = \begin{cases} h_{min}, & 0 \leq r \leq R, \\ ar^{\alpha}, & R \leq r \leq R_o, \\ h_{max}, & R_o \leq r, \end{cases} \tag{2.9}$$

where r – the distance from the center of cable, R_o – the outer radius of adaptive area, R – the inner radius of adaptive area, h_{min} – the size of triangles in the cable, h_{max} – the size of triangles in the subdomain with soil. a and α are calculated from conditions

$$\begin{cases} aR^{\alpha} = h_{min}, \\ aR_o^{\alpha} = h_{max}, \end{cases} \tag{2.10}$$

$$\begin{aligned} a &= R^{-\ln\left(\frac{h_{max}}{h_{min}}\right)(-\ln(R)+\ln(R_o))^{-1}} h_{min}, \\ \alpha &= \ln\left(\frac{h_{max}}{h_{min}}\right)(-\ln(R)+\ln(R_o))^{-1}. \end{aligned} \tag{2.11}$$

For all experiments $h_{max} = 10h_{min}$ and $R_o = 2R$, where R is the radius of the investigated cable. The adaptive mesh is illustrated in Fig. 2.19. The data in Table 2.2 is also obtained using adaptive mesh. As it was mentioned before, the results presented in Table 2.2 show the convergence rate close to 2 compared to convergence rate equal to 1 for meshes that were generated using OpenFoam built-in tools (see Table 2.1).

An algorithm to control mesh singularities. Consider an example where two cables touch each other or the cable is inside the circular domain and is touching its boundary. During mesh generation circular domains are approximated by edges (linear), so cables are not connected when the geometry is approximated by a mesh. The distance between cables can be very small, so it can raise numerical problems or even a floating point exception. Such a situation is called a mesh singularity. It is necessary to avoid such singularities that is why the automatic singularity controlling algorithm is proposed in this section.

The idea of our algorithmic approach is to identify the positions of singularities and define the special edges that the contours will be forced to lay

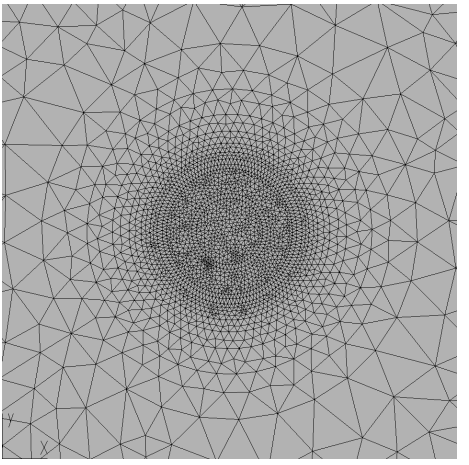


Fig. 2.19. The mesh with a priori controlled adaptivity

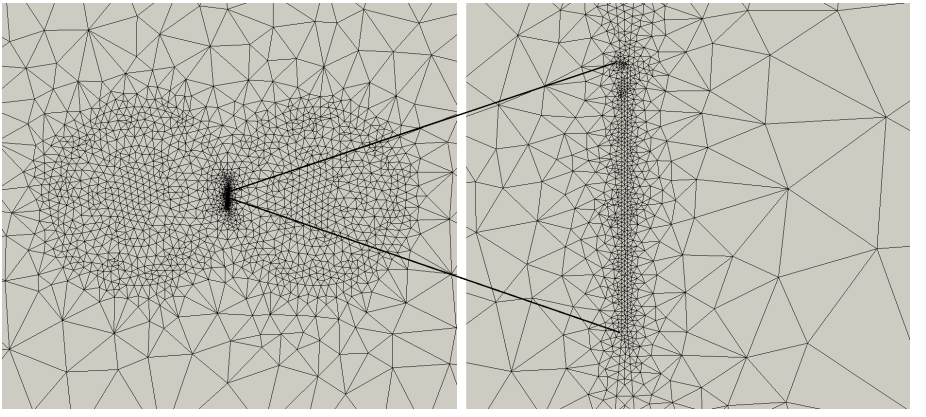


Fig. 2.20. The singularity of a mesh

on these edges. Let these edges to be called bridges. Let N be the number of triangles, M – the number of circles, R_i – the radius of ith circle, C_i – the center of ith circle, B – the vector of bridges.

Denote a function $makeBridge(C, V)$ which creates the edge with center at coordinates C, normal vector V and length $C_h h$, where C_h is a constant, h is the size of triangles. The following algorithm to identify and construct the bridges is proposed:

Algorithm 4 Construction of bridges in the mesh

$B = \oslash$
for $i = 1$ to M **do**
  for $j = 1$ to M **do**
    if $i > j$ and $|C_j - C_i| = R_i + R_j$ **then**
      $B.add\left(makeBridge\left(C_i + \frac{C_j - C_i}{|C_j - C_i|}R_i, \frac{C_j - C_i}{|C_j - C_i|}\right)\right)$
    end if
    if $i\ != j$ and $|C_j - C_i| = R_i - R_j$ **then**
      $B.add\left(makeBridge\left(C_i + \frac{C_j - C_i}{|C_j - C_i|}(R_i - R_j), \frac{C_j - C_i}{|C_j - C_i|}\right)\right)$
    end if
  end for
end for

To force the edges to lay on bridges the Algorithm 5 is used.

Algorithm 5 The modification for generation of edges

for $i = 1$ to M **do**
  while $E = makeNewEdge(i)$ **do**
    for $\forall b \in B$ **do**
      if E is near B **then**
        $addBridgeToGraph(b)$
        while E is near B **do**
          $E = makeNewEdge(i)$
        end while
      end if
    end for
    $addEdgeToGraph(E)$
  end while
end for

The example of generated mesh with singularity control is presented in Fig. 2.21.

Lets analyze the additional computational costs for this modification of a mesh generation. Algorithm 4 takes $O(M^2)$ operations to complete, computational costs in Algorithm 5 are equal to $O(M\sqrt{N}S)$, where S is the number of singularities and assumption was made that the number of edges on con-

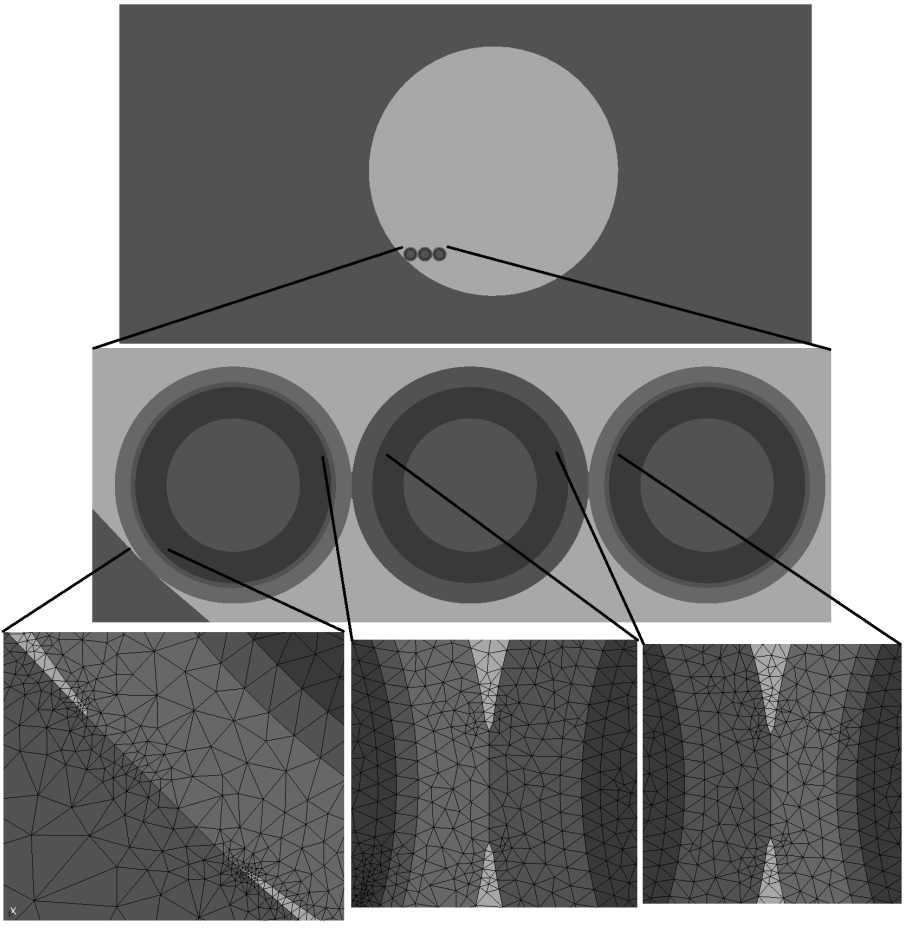


Fig. 2.21. The example of a mesh when circles touch each other in different ways

tours of circles is equal to $O\sqrt{N}$. Assuming that the number of circles M is negligibly small comparing to $\sqrt{N}$, the overall complexity of algorithm is $O(M\sqrt{N}S)$. Note that aCute software generates a mesh using $O(N)$ operations, so to guarantee that the modification introduces computations of the same order we should have $MS = O(\sqrt{N})$, then $O(M\sqrt{N}S) = O(N)$. The conclusion from this analysis is that taking $MS = O(\sqrt{N})$ gives no significant effect on mesh generation time, this was confirmed experimentally. For real problems $MS \approx 100$, so applying the mesh singularity controlling

approach have no significant effect on the total mesh generation times, for meshes used in real computational experiments with $N \geq 10^5$.

Numerical experiments. We define the heat source density function as:

$$q = \varrho J^2,$$

where ϱ is resistivity of conductor, J – electric current density measured in A/m^2. Taking $\varrho = \varrho_0(1 + \alpha(T - T_0))$ gives

$$q = \varrho_0(1 + \alpha(T - T_0))J^2. \tag{2.12}$$

Here α is temperature resistance coefficient of core, ϱ_0 is it is resistivity at temperature $T = T_0$. The problem is solved for 24 hours, so $t \in [0, 24]$. The temperature source function is defined as follows:

$$q = \begin{cases} g_0(T)f_1(t), x \in \Omega_{1,m}, \\ g_0(T)f_2(t), x \in \Omega_{2,m}, \\ g_0(T)f_3(t), x \in \Omega_{3,m}, \end{cases} \tag{2.13}$$

where $\Omega_{i,m}$ is the metal domain of i-th cable (the numeration by index i starts from the left side – see Fig. 2.5). Functions g_0, f_i are defined as:

$$g_0 = 75105(1 + 0.00393(T - 293.15)), \tag{2.14}$$

$$f_1 = \begin{cases} 1, t \leq 12, \\ 0, t > 12, \end{cases} \tag{2.15}$$

$$f_2 = \begin{cases} 0.3, t \leq 6, \\ 1.3, 6 < t \leq 8, \\ 1, 8 < t \leq 18, \\ 1.3, 18 < t \leq 20, \\ 0.3, t > 20, \end{cases} \tag{2.16}$$

$$f_3 = 0.5\left(2 + \sin\frac{\pi}{2}\right), \tag{2.17}$$

the functions $f1(t)$, $f1(t)$ and $f1(t)$ are presented graphically in Fig. 2.22.

The problem was solved with $N = 6018$ and with a fixed time step $\tau = 0.5$ hours. The corresponding calculations took around 3 seconds with a

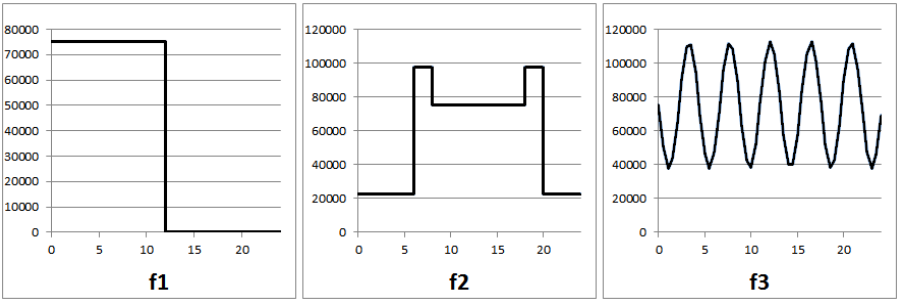


Fig. 2.22. Heat source functions

single core AMD Phenom™ II X6 1100T 3.3 GHz processor, big part computations time it was initialization of OpenFOAM computational routines. Time step is taken big enough to make visual results representative. The coordinates centre of the cables is assigned as x_{C_i}, where i is the cable number. The corresponding dynamical functions of temperature in the centres of the cables $T(x_{C_i}, t)$ are presented in Fig. 2.23, Fig. 2.24, Fig. 2.25.

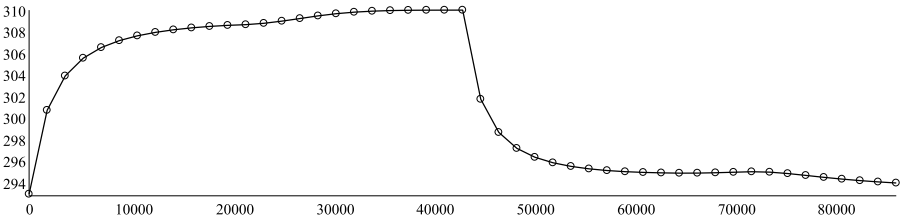


Fig. 2.23. The temperature in the centre of the first cable $T(x_{C_1}, t)$

As it is seen from these figures, cables affect each other, despite the fact that they are quite far from each other:

- after 20 hours the temperature in the cable 1 becomes smaller when the temperature of cable 2 decreases;
- the temperature in the cable 2 decreases, after the source for the cable 1 becomes 0;
- the temperature in the cable 3 is smaller in the local minimum after 22 hours than in the local minimum before 20 hours, since the temperature in cable 2 decreases greatly.

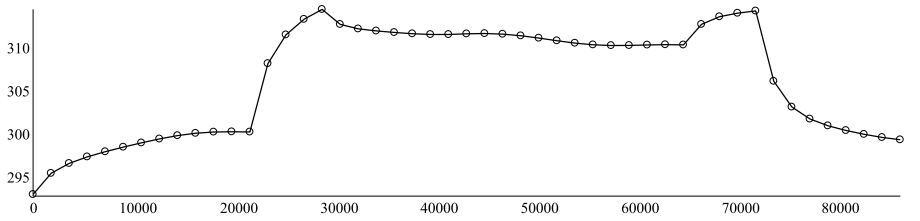


Fig. 2.24. The temperature in the centre of the second cable $T(x_{C_2}, t)$

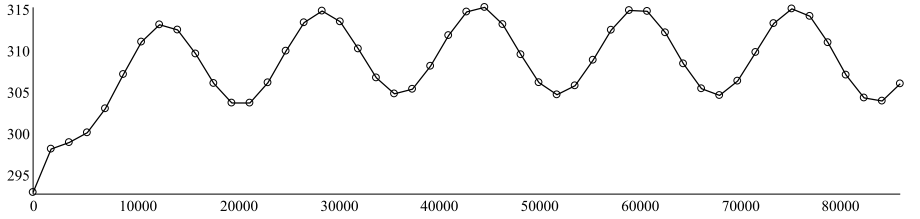


Fig. 2.25. The temperature in the centre of the third cable $T(x_{C_3}, t)$

Also, it is possible to note that after big changes of the source function all solutions are changing fast, so the usage of adaptive time step strategy would be reasonable. However, time-dependent error solving strategy and adaptive time step selection are not discussed in this work.

2.4. Introduction Into Parallel Computing Using OpenFOAM

Presently applicable IEC standards for the design and installation of electrical power cables are often based on the analytical and heuristic formulas.Obviously, these formulas cannot accurately account for the various conditions under which the cables are actually installed and used. They estimate the cable's current-carrying capacity (so-called *ampacity*) with significant margins to stay on the safe side (Makhkamova 2011). The safety margins can be quite large and result in 50–70% usage of actual resources. A more accurate mathematical modelling is needed to meet the latest technical and economical requirements and to elaborate new, improved, cost-effective design rules and standards.

When we need to deal with mathematical models for the heat transfer in

various media (metals, insulators, soil, water, air) and non-trivial geometries, only the means of parallel computing technologies can allow us to get results in an adequate time. To solve numerically selected models, we develop our numerical solvers using the OpenFOAM package (OpenFOAM 2015). OpenFOAM is a free, open source CFD software package. It has an extensive set of standard solvers for popular CFD applications. It also allows us to implement our own models, numerical schemes and algorithms, utilizing the rich set of OpenFOAM capabilities (Weller, Tabor, Jasak, Fureby 1998). Adapting OpenFOAM library to specific applications still requires theoretical analysis of selected algorithms and nontrivial selection of optimal data structures for the implementation of required algorithms. Examples of such projects are described in (Higuera, Lara, Losada 2013; Petit, Bosioc, Nilsson, Muniean, Susan-Resigo 2011).

The important consequence of this software development approach is that numerical solvers can automatically exploit the parallel computing capabilities already available in the OpenFOAM package. A detailed analysis on implementation of some types of parallel algorithms for GPU processors is done in (AlOnazi 2013).

Scalability and performance of parallel OpenFOAM solvers based on MPI for various applications are investigated in (Piscaglia, Montorfano, Onorati 2013; Rivera, Furlinger, Kranzimuller 2011). Computational experiments are done on homogeneous distributed parallel platforms with up to 1024 cores. It is noted in (Rivera, Furlinger, Kranzimuller 2011) that the scalability and efficiency of parallel OpenFOAM solvers is not very well understood for many applications when executed on massively parallel systems. An extensive experimental analysis of OpenFOAM selected applications is done in Prace project. A few CFD applications with different multi-physics models are approximated by FVM on mainly fully structural 3D meshes. Mesh partition is done by using Simple and Scotch tools. The presented experimental results are showing a good OpenFOAM scaling and efficiency performance on IBM Blue Gene Q and Hewlett Packard C7000 parallel systems up to 2048- 4096 cores. It is noted that such results are expected when computations, message passing and I/O are well balanced.

In this work, we study and analyze the parallel performance of OpenFOAM-based solver for heat conduction in electrical power cables. The main goal is to consider the scalability and efficiency of the developed parallel solver in the case when the parallel system is not big, but it consists of non homogeneous multicore nodes. The mesh is adaptive and it is partitioned using Scotch tool. The load balancing techniques must be used in order to opti-

mize the parallel efficiency of the solver. The second aim is to investigate the sensitivity of parallel preconditioners with respect to the number of processes.

In Section 2.5, we describe the benchmark problem used for all numerical tests. In Section 2.6, we describe our OpenFOAM-based solver and discuss the parallelization approach employed in the OpenFOAM package. The theoretical scalability analysis of the parallel algorithm is presented in Section 2.7. In Section 2.8, we present and analyze the obtained results on parallel efficiency and scalability of the solver. Finally, some conclusions are done in Section 2.9.

2.5. Benchmark Problem

As a benchmark problem in this research we solve the heat conduction problem for electrical power cables directly buried in the soil. It is also assumed that the thermo-physical properties of the soil remain constant, i.e. the moisture transfer in the soil is not considered. Such a simplified problem is described by the well-known mathematical model that was presented before (2.1).

Their values can vary between metallic conductor, insulators and soil by several orders of magnitude (Makhkamova 2011).

For our benchmark problem we model three cables that are buried in the soil as shown in Fig. 2.5. The red area is metallic conductor, the blue area is an insulator and the gray area marks the soil.

OpenFOAM (Open source Field Operation And Manipulation) (OpenFOAM 2015) is a C++ toolbox (library) for the development of customized numerical solvers for partial differential equations (PDEs). For benchmark problem (2.1) we obtain a numerical solver by a modification of the standard *laplacianFoam* solver, adding variable problem coefficients. OpenFOAM uses the Finite Volume Method (FVM) with co-located arrangement of unknowns (Weller, Tabor, Jasak, Fureby 1998).

Two important sub-tasks should be solved accurately for this type of approximations. First, the exact fluxes of a solution are orthogonal to the boundary of finite volumes, thus for numerical fluxes this property must be approximated accurately. We note that OpenFOAM tool proposes iterative interpolation type orthogonalization techniques however that lowers the optimal rate of convergence (see Table 2.1). In our solver this problem is solved by using a proper Delaunay triangulation of the domain (see Table 2.2). Second, a proper interpolation should be used for definition of discontinuous coefficients λ in Laplacian term, namely *harmonic*, this possibility is included in OpenFOAM.

Then for the 2D benchmark problem (2.1) by using Delaunay type triangu-

lation we obtain FVM discretization with the four point stencil. In 3D case the uniform mesh is applied in the additional third dimension and the three-point stencil is used to approximate the fluxes in this direction. Resulting systems of linear equations with symmetric matrices are solved by preconditioned conjugate gradient method with the diagonal Incomplete-Cholesky (DIC) preconditioner.

2.6. Parallel OpenFOAM-based Solver

Parallelization in OpenFOAM is robust and implemented at a low level using MPI library. Solvers are built using high level objects and, in general, do not require any parallel-specific coding. They will run in parallel automatically. Thus there is no need for users to implement standard steps of any parallel code: decomposition of the problem into subproblems, distribution of these tasks among different processes, implementation of data communication methods. A drawback of such automatic tools is that the user has very limited possibilities to modify the generated parallel algorithm if the efficiency of the OpenFOAM parallel code is not sufficient.

OpenFOAM employs a common approach for parallelization of numerical algorithms – domain decomposition. The mesh and its associated fields are partitioned into sub-domains, which are allocated to different processes. Parallel computation of the proposed finite FVM algorithm requires two types of communication: local communications between neighboring processes for approximation of the Laplacian term on the given stencil and global communications between all processes for computation of scalar products in DIC iterative method.

OpenFOAM employs a zero-halo layer approach (AlOnazi 2013), which considers cell edges on sub-domain boundaries as boundary and applies a special kind of boundary condition. Such approach minimizes ammount of data communicated among processors but the convergence rate can be not optimal.

OpenFOAM supports four methods of domain decomposition, which decompose the data into non-overlapping sub-domains: simple, hierarchical, scotch and manual (OpenFOAM 2015). In all parallel tests the mesh is partitioned by using Scotch library (Chevalier, Pellegrini 2008). Scotch is a library for graph and mesh partitioning, similar to well-known Metis library (Karypis 2015). It requires no geometric input from the user and attempts to minimize the number of boundary edges between sub-domains. The user can specify the weights of the sub-domains, what can be useful on heterogeneous clusters of

parallel computers with different performance of processors. We will use this possibility in our computational experiments.

2.7. Scalability Analysis of the Parallel Algorithm

In this section we are interested to investigate the efficiency of the parallel algorithm generated by the OpenFOAM tool with respect to load balancing and data communication costs and by. Thus in all numerical tests (and in the scalability analysis) we compute 10 time steps, number of iterations for solving systems of linear equations is fixed to 1000. In this way, we ensure that the same amount of work is done in all parallel tests, despite the possible differences in convergence due to parallel preconditioning and different roundoff errors, including data communication subroutines. The influence of mesh partitioning on parallel preconditioners is investigated in the last subsection of computational experiments.

Let us estimate the costs of the sequential algorithm to compute a solution at one time step as

$$W = (c_1 + 1000c_2)J,$$

where J is the total number of finite volumes in the mesh, c_1 estimates the costs of computation of the coefficients of the discrete system, c_2 estimates the costs of one DIC iteration.

Next let us estimate the complexity of the proposed parallel algorithm. For simplicity of theoretical scalability analysis let us assume that p homogeneous processes are used in computations. Then computation costs of the parallel algorithm can be estimated as

$$T_p^1 = (c_1 + 1000c_2)\lceil J/p \rceil.$$

Two assumptions have been used in derivation of this estimate. First, we have assumed that a perfect load balancing of sizes of partitioned mesh parts is achieved, this assumption usually is very accurately satisfied for meshes partitioned by Metis or Scotch libraries. Second, we are not taking into account that $c_1 = c_1(p)$, $c_2 = c_2(p)$ may depend on p and they can decrease for $p > 1$ due to a better cashing of smaller size discrete subproblems. In fact, such a behaviour will be illustrated by results of computational experiments. In the theoretical scalability analysis we are considering the worst case scenario.

Next we will estimate costs of communication among processors. As was stated above the implementation of the given parallel algorithm requires local

send/receive of data between neighbour processes and global communication in computation of scalar products for Krylov type iterations. We assume that the largest number of data items sent between neighbour processes can be estimated as $c_3\sqrt{J}$ and let M be the largest number of neighbours for some process. Then the communication costs can be estimated as (Kumar, Grama, Gupta, Karypis 1994)

$$T_p^2 = 1000\big[r(M)(\alpha + \beta c_3\sqrt{J}) + R(p)\big].$$

Here α denotes the message startup time and β is the time required to send one element of data. Coefficients $1 \le r(M) \le M$ and $\log p \le R(p) \le p$ define the parallel efficiency of local data exchange and global reduce operations. The values of $r(M)$ and $R(p)$ depend on the implementation of MPI functions and on interconnection network of a parallel computer. For example, for a simple implementation of `MPI_ALLREDUCE` function, when all processors send their local values to the master processor, which accumulates results and broadcasts the sum to all processors, $R(p) = p$. On the 2D mesh network this function can be implemented with $R(p) = c\sqrt{p}$.

Thus the total complexity of the parallel algorithm is equal to

$$T_p = (c_1 + 1000c_2)\lceil J/p\rceil + 1000\big[r(M)(\alpha + \beta c_3\sqrt{J}) + R(p)\big].$$

The scalability analysis of any parallel algorithm finds the rate at which the size of the sequential algorithm W needs to grow up with respect to the number of processes p in order to maintain a fixed efficiency of the parallel algorithm $E_p = W/(pT_p)$. For a given efficiency E the isoefficiency function $W = g(p, E)$ is defined by the implicit equation (Kumar, Grama, Gupta, Karypis 1994):

$$W = \frac{E}{1-E}\,H(p, W)\,. \tag{2.18}$$

The total overhead of the proposed parallel algorithm is given by

$$\begin{aligned} H(p, W) &:= pT_p - W \\ &= (c_1 + 1000c_2)\big(p\lceil J/p\rceil - J\big) + \\ &\quad 1000p\big[r(M)(\alpha + \beta c_3\sqrt{J}) + R(p)\big]. \end{aligned}$$

Let us assume that the effects of load disbalance and start-up time of communications are negligible. Then it follows from (2.18) that asymptotical isoeffi-

ciency functions due to local and global communications both have the same order $W = O(p^2)$.

Table 2.4. Analysis of the mesh decomposition algorithm: M_p denotes the largest number of neighbours, N_p denotes the maximum number of elements communicated between two processes and NT_p denotes the largest total number of elements sent by some process

J	32000	128000	512000	1018488	2048000
M_2	1	1	1	1	1
N_2	208	442	1386	3011	4854
M_4	3	2	3	3	3
N_4	148	466	1038	2383	3057
NT_4	342	826	2348	4023	6752
M_8	5	5	7	7	7
N_8	279	279	836	1224	2395
NT_8	681	681	2432	3851	5915

In Table 2.4 we present a basic information on the quality of the mesh decomposition algorithm. The load balancing of sizes of subproblems was very close to optimal, thus we restrict to analysis of data communicated among processes. Here M_p denotes the largest number of neighbours for some process, N_p denotes the maximum number of elements communicated between two processes and NT_p denotes the largest total number of elements sent by some process to its neighbours.

The results in Table 2.4 show that $O(\sqrt{J})$ is also a realistic estimate of the total number of elements sent by one process to its neighbours and the dependence of this number on p is very weak.

2.8. Parallel Performance Tests and Analysis

The computations were done on the Vilkas cluster at Vilnius Gediminas technical university. It consists of eight Intel Quad i7-860 processors with 4 cores (2.80 GHz) per node and eight Intel Quad Q6600 processors with 4 cores (2.4 GHz) per node. They are interconnected via Gigabit Smart Switch (http://vilkas.vgtu.lt). We also note that the sub-cluster of Intel Quad i7-860 processors is not fully homogeneous. Thus the Vilkas cluster is quite heterogeneous and therefore additional weighted load balancing is included into mesh distribution step.

In Table 2.5 we present CPU times of computational experiments with different nodes of the cluster. Here $i7$-x and q-x denote the x-th Intel Quad i7-860 and Intel Quad Q6600 processor, respectively. The number of iterations for solving systems of linear equations by using DIC preconditioner is fixed to 1000. In the case of Quad Q6600 nodes we have presented results only for the fastest and slowest nodes.

Table 2.5. CPU times of the sequential algorithm for different sizes of the problem and different processors of Vilkas cluster: $i7$-x and q-x denote the x-th Intel Quad i7-860 and Q6600 processor, respectively.

J	128000	254892	512000	1018488	2048000
$i7$-0	36.7	82.7	201.9	415.9	909.2
$i7$-1	36.6	82.9	202.0	415.5	893.3
$i7$-2	36.7	82.4	198.6	413.0	887.9
$i7$-3	35.2	77.7	177.0	362.3	772.0
$i7$-4	36.8	82.4	198.3	415.8	887.6
$i7$-5	36.8	83.6	199.0	412.6	874.4
$i7$-7	36.1	81.1	190.7	390.2	839.3
$i7$-8	36.9	83.2	200.3	417.8	868.1
q-8	66.2	151.3	336.5	677.5	1441
q-13	66.3	153.9	341.3	678.2	1442

Two important conclusions follow from the presented computational results. First, due to memory cashing effects, the CPU time of the OpenFOAM solver increases over-linearly with respect to the size J of the discrete problem. On the basis of these experimental results we propose the following computation time prediction model for the parallel OpenFOAM solver

$$T_p^1(J) = \max_{x \in G} T_0(x, J/p), \tag{2.19}$$

where G is set of p processors used to solve the problem of size J and $T_0(x, J/p)$ denotes the CPU time of the sequential algorithm applied for problem of size J/p on the x-th processor.

The second conclusion is that Intel Quad i7-860 processors are approximately 1.6 times faster than Q6600 processors. In addition some Intel Quad i7-860 processors are up till 1.15 times faster than the remaining processors. Thus a weighted load balancing technique can reduce the global CPU time of the parallel solver.

Next we present results of computational experiments, which confirm both conclusions. In Table 2.6 CPU times of the parallel OpenFOAM algorithm are

given for different sizes of the discrete problem and different sets of processors.

Table 2.6. CPU times of the parallel OpenFOAM algorithm for different sizes of the problem and different sets of processors: $i7$-x and q-x denote the x-th Intel Quad i7-860 and Q6600 processor, respectively.

J	128000	254892	512000	1018488	2048000
$i7$-0, $i7$-1	18.8	39.3	86.7	204.9	420.2
$i7$-3, $i7$-7	18.4	38.3	86.1	193.3	394.4
q-8, q-9	30.7	73.2	162.0	337.5	689.5
$i7$-3, q-8	25.5	67.41	155.4	331.6	683.6
$i7$-0, $i7$-1 $i7$-2, $i7$-4	11.8	20.4	41.3	89.4	209.4
$i7$-3, $i7$-5 $i7$-7, $i7$-8	11.4	20.4	40.9	90.0	204.3
q-8, q-12 q-9, q-13	23.4	37.9	79.4	168.5	347.6
8 $i7$ nodes	8.8	13.6	22.4	43.6	94.3
8 q nodes	22.1	28.0	44.3	86.7	179.5
16 nodes	17.3	20.5	26.2	42.7	85.4

It follows from the presented results that for two largest size discrete problems the CPU time is shorter when all 16 processors of the cluster are used.

Since Vilkas cluster is heterogeneous and consists of two types of nodes, a load balance of computational tasks can be improved by using the weighted mesh partitioning algorithm. In Table 2.7 CPU times of the parallel OpenFoam algorithm are given for different relative weights assigned to processors. Here $i7(w)$ and $q(w)$ denote Intel Quad i7-860 or Q6600 processors and w denotes the relative speed of this processor.

It follows from these results that adaptive mesh distribution algorithm improves the load balancing and CPU time decreases 1.3 times for the largest discrete problem.

Next we have investigated the efficiency of the parallel solver when 2 and 4 cores per node are used in computations. The first conclusion is that only 2 cores are giving a reasonable speed-up of computations. Thus in Table Table 2.8 we present CPU times of the parallel OpenFoam algorithm for different numbers of processors and 2 cores per node. The size of the problem is $J = 2048000$ elements. The case 1×1 provides CPU time for the sequential algorithm.

Table 2.7. CPU times of the parallel OpenFoam algorithm for the adaptive mesh decomposition algorithm. Here $i7(w)$ and $q(w)$ denote Intel Quad i7-860 or Q6600 processors and w denotes the relative speed of this processor

J	512000	1018488	2048000	8192000
$i7$-3(1), q-8(1) q-9(1)	102.0	222.0	461.1	
$i7$-3(2), q-8(1) q-9(1)	89.7	189.9	381.3	
$i7$-3(1.87), q-8(1) q-9(1)	88.8	183.0	371.8	
8 $i7$(1), 8 q(1)	26.2	47.7	85.4	363.6
8 $i7$(1.5), 8 q(1)	24.3	35.5	69.4	288.2
8 $i7$(1.6), 8 q(1)	24.5	35.7	66.1	279.9

Table 2.8. CPU times of the parallel OpenFoam algorithm for different numbers of processors n_d and $n_c = 2$ cores per node. The size of the problem is $J = 2048000$ elements. The case 1×1 provides CPU time for the sequential algorithm

	1×1	1×2	2×2	4×2	8×2
$i7$-3	772.0	566.5	288.2	142.7	64.8
q-9	1441	1161	573.6	270.2	122.7

Two conclusions follow from the presented results. First, the scalability of the parallel algorithm is still good for clusters with multicore nodes. This scaling follows very similar trends as for one core per node. The second conclusion states that the retardation coefficient $\mu(n_c) > 1$ should be included into the estimate of computation costs of the parallel algorithm

$$T_p^1 = (c_1 + 1000\mu(n_c)c_2)\lceil J/p \rceil.$$

It depends on the maximum number of cores n_c per processor. This coefficient should be taken into account, since the shared-memory structure can become a bottleneck when too many cores try to access the global memory of a node simultaneously.

In the next series of computational experiments we have solved the largest problem with $J = 8192000$ elements. 16x2 processes of two types of nodes were used in computations. Since Vilkas cluster is heterogeneous, a load balancing strategy is applied in the weighted mesh partitioning algorithm. In Table 2.9 CPU times of the parallel OpenFoam algorithm are given for different relative weights assigned to processors.

Table 2.9. CPU times of the parallel OpenFoam algorithm for the adaptive mesh decomposition algorithm. Here $i7(w)$ and $q(w)$ denote Intel Quad i7-860 or Q6600 processors and w denotes the relative speed of this processor. The size of the discrete problem $J = 8192000$

	$i7(1), q(1)$	$i7(1.4), q(1)$	$i7(1.5), q(1)$	$i7(1.6), q(1)$	$i7(1.7), q(1)$
T_{32}	279.9	229.7	216.6	206.3	202.4

Up to this point all results were obtained by fixing the number of linear solver iterations to 1000. In practice the number of iterations is calculated dynamically to fit the convergence tolerance requirement. Since we are using the conjugate gradient method with the diagonal Incomplete-Cholesky (DIC) preconditioner, the number of iterations may depend on the number of processes p. For computational tests we take $J = 2048000$. Results of computational experiments show that the time of computations is proportional to the number of iterations and the number of iterations weakly depends on the number of processes. So for different numbers of processes p we calculate the number of iterations that are performed to achieve the tolerance equal to 10^{-6}.

Table 2.10. The average number of iterations per time step for different number of processes p. The size of the discrete problem $J = 2048000$, the tolerance parameter for solver is equal to 10^{-6}

	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$
Av. num. of it.	1015.6	1140.1	1142.6	1448.2	1401.6
T_p	803.7	452	236.1	135.6	91.5

The results in Table 2.10 show that the increased number of iterations lowers the efficiency, this number can occasionally also drop. But in general efficiency of parallel preconditioners is not sensitive to changes of p. In the case of $p = 16$ processes we have used 8 nodes and 2 cores per node.

2.9. Conclusions of the Second Chapter

1. Two different strategies of generating spatial meshes for heat transfer simulation in underground electrical cables were discussed in this work. Meshes that were generated fit geometry of subdomains contours.
2. Usage of rectangular meshes and non-orthogonality corrections methods built-in into OpenFOAM gave a reduced non-optimal order of convergence of FVM solution.
3. Domain triangularization with cell centres in Voronoi points eliminates non-orthogonality errors, keeping the optimal FVM order of convergence.
4. For solving real industrial problems spatial adaptive mesh is essential and adaptive time steps also may be needed.
5. Heuristic for generating adaptive spatial mesh is proposed. Finally, harmonic mean formulas deal with big coefficients jumps, keeping the error small enough for modelling purposes.
6. The analysis of proposed algorithm to control mesh singularities shows that for considered cases it solves the efficiency problem (when mesh adaptivity is essential).
7. The problem is solved using distributed calculations. It is shown that for a developed OpenFOAM solver the scaling and efficiency performance on Vilkas cluster is good up to 32 cores when I/O effects are neglected and load balancing is used for mesh partition.
8. Smaller sizes of distributed discrete sub-problems enable a better caching and give a sub-linear speed-up for computational part of the parallel algorithm.
9. It is important to test the effects of I/O costs when balancing between computation and I/O parts of the algorithm is not good, for example when a solution should be saved every 5–10 time steps.
10. It is well known that OpenFOAM I/O libraries are based on standard MPI I/O routines and they are introducing quite big overheads. A hybrid MPI and OpenMP parallel model can be attractive in the case of parallel systems with a big number (12 or 16) of cores per node.

3

The Methods for Non-stable Physical Process Models

In this chapter it is discussed how to model the process that forms a stochastic pattern (Eisenbach 2004). It is shown that chemotaxis-driven instability can be connected to the ill-posed problem defined by the backward in time diffusion process. By using the well-established techniques the mathematical model is approximated by the discrete computational model. It is proved that the discrete solution inherits main properties of the solution of the differential mathematical model. An algorithm for distributed calculations is proposed and analyzed.

Parts of this chapter are published in (Čiegis, Bugajev 2013) and (Čiegis, Bugajev 2014).

3.1. Introduction Into Third Chapter

In this chapter a special case of modelling will be observed, when the implementation itself can make a strong impact on results due to sensitivity of a given model. That means that the formulated model can correctly describe all physical process and this model belongs to some general class of models, however, when standard (for this class of models) methods are applied and implemented – it becomes clear that the results are very sensitive to almost any part of modelling scheme: initial data, round off errors and even methods osed to obtain the solution. That raises the question on how do solvers should be implemented in order to obtain the results that are robust and correctly describe important properties of the real physical process. Another important question is how do such problems should be solved using distributed calculations.

The problem presented in this chapter is solved using finite difference approximations. Mathematical model of a bacterial self-organization is considered. In article "Parallel algorithms for three-dimensional parabolic and pseudoparabolic problems with different boundary conditions" we show how to approximate similar problems in more general case – when the problem is three-dimensional and have different types of boundary conditions in different directions. In mentioned paper three-dimensional parabolic and pseudoparabolic equations with classical, periodic and nonlocal boundary conditions are approximated by the full approximation backward Euler method, locally one dimensional and Douglas Alternating Direction Implicit (ADI) splitting schemes. The stability of these methods are investigated. Note that parabolic and nonlocal boundary conditions require very special modifications of algorithms, including paralle algorithms. As was mentioned above, stability with respect to boundary condition can be one of properties defining the global stability/unstability of the full model.

However, as it was mentioned before, in this chapter the model that have the unstability as the main issue is investigated. The dynamics of such nonlinear systems can lead to formation of complicated solution patterns. It was shown show that this chemotaxis-driven instability can be connected to the illposed problem defined by the backward in time diffusion process. The method of lines is used to construct robust numerical approximations. At the first step we approximate spatial derivatives in the PDE by applying approximations targeted for special physical processes described by differential equations. The obtained system of ODE is splitted into a system describing separately fast and slow physical processes and different implicit and explicit numerical solvers

are constructed for each subproblem. Results of numerical experiments are presented and convergence of finite difference schemes is investigated.

Chemotaxis. Many microscopic organisms must move by responding to chemicals in order to survive. The directed movement of microorganisms in response to chemical gradients is called chemotaxis (Eisenbach 2004). Chemotaxis is very important in a wide range of biological phenomena. Within the embryo, chemotaxis affects avian gastrulation and patterning of the nervous system. The same mechanisms are observed during cancer growth, allowing tumour cells to invade into healthy tissue or drive new blood vessel growth (angiogenesis).

Baronas, Šimkus (2011) has considered the pattern formation in a luminous Escherichia coli bacteria colony. He considered a nonstationary one-dimensional-in-space mathematical model of a bacterial selforganization in a circular container along the contact line as detected by quasi-one-dimensional bioluminescence imaging. Fig. 3.1 shows typical top view bioluminescence images of bacterial cultures illustrating an accumulation of luminous bacteria near the contact line. In general, the dynamic processes in unstirred cultures are rather complicated and need to be modeled in three dimensional space. Nevertheless, the accumulation of luminous cells near the contact line implies that the essentially three-dimensional processes may be approximated in one dimension (quasi-one dimensional rings in Fig. 3.1). The typical space-time plot of quasi-one-dimensional bioluminescence intensity is shown in Fig. 3.2.

If we talk about the size of computations then one-dimensional problem is trivial and can be easily solved using an average personal computer. However, we use the one dimensional model to analyze the problem and to develop methods that can be generalized to be applied two-dimensional model. Later we generalize these methods to solve two dimensional problem and modify it to develop parallel algorithm version.

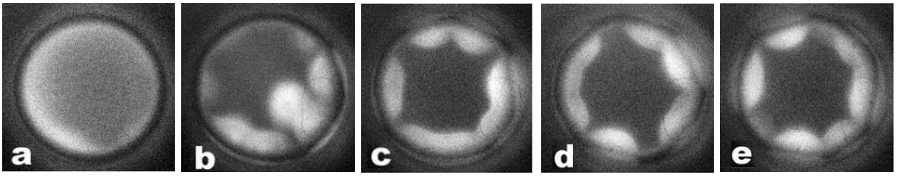


Fig. 3.1. Top view bioluminescence images of the bacterial cultures in the cylindrical vessel at different time moments: (a) 5, (b) 20, (c) 40, (d) 60 and (e) 90 min (Šimkus, Kirejev, Meškienė, Meškys 2009)

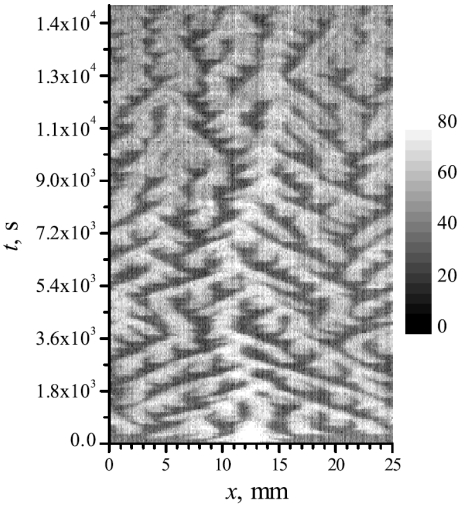


Fig. 3.2. Space-time plot of bioluminescence measured in a circular container along the contact (Baronas, Šimkus 2011)

Problem. It is well-known that many mathematical problems of biological systems are described by non-stationary and non-linear diffusion-advection-reaction equations. The dynamics of their solutions can be very complicated, the interaction of different physical processes can lead to development of spatial and temporal patterns and instabilities (Murray 2002). In addition, solutions of biological and chemical systems satisfy special properties, such as positivity, boundedness and conservativity (Hundsdorfer, Verwer 2003). Therefore the development of robust and efficient numerical algorithms for solution of such problems still remains a very important challenge for computational engineers.

Consider a mathematical model for simulation of the bacterial self-organization in circular container along the contact line, which is proposed by Baronas, Šimkus (2011). An excellent review on PDE models for chemotaxis was made by Hillen, Painter (2009).

By using some simplifying assumptions, the mathematical model is de-

fined as a system of one dimensional nonlinear equations:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - \frac{\partial}{\partial x}\left(\frac{\chi u}{(1+\alpha v)^2}\frac{\partial v}{\partial x}\right) + \gamma r u(1-u), \tag{3.1}$$
$$\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2} + \gamma\Big(\frac{u^p}{1+\beta u^p} - v\Big), \quad x \in (0,1), \;\; t > 0,$$

where u is the dimensionless cell density, v is the dimensionless chemoattractant concentration, α defines the receptor sensitivity, β stands for saturating of the signal production, γ defines the ratio of the spatial and temporal scales, and $p \geq 1$. The initial conditions are defined as

$$u(x,0) = u_0(x), \quad v(x,0) = 0, \quad x \in [0,1]. \tag{3.2}$$

The periodicity conditions are formulated as boundary conditions:

$$u(0,t) = u(1,t), \quad \frac{\partial u}{\partial x}\Big|_{x=0} = \frac{\partial u}{\partial x}\Big|_{x=1}, \tag{3.3}$$
$$v(0,t) = v(1,t), \quad \frac{\partial v}{\partial x}\Big|_{x=0} = \frac{\partial v}{\partial x}\Big|_{x=1}.$$

Analysis of the Mathematical Model The non-deterministic initial conditions are investigated by Baronas, Šimkus (2011):

$$u(x,0) = 1 + \varepsilon(x), \quad x \in [0,1],$$

where $\varepsilon(x)$ is a 20% random uniform spatial perturbation. On the basis of computational simulations it is demonstrated that solutions of the proposed mathematical model can develop complicated spatial-time patterns observed also in real bioluminescence images (Brenner, Levitov, Budrene 1998; Budrene, Berg 1995).

Note that random perturbation of initial data is not a necessary condition for self-organization of the solution. The sensitivity of the solution with respect to initial data should be investigated, i.e. to analyze the well-posedness of the mathematical model. Many technological and physical processes can lead to development of spatial and temporal instabilities in solutions, we mention only a fingering instability in thin evaporating liquid films (Cazabat, Heslot, Troian, Carles 1990; Lyushnin, Golovin, Pismen 2002), in buoyancy-driven fluid filled cracks (Touvet 2011) or in porous media flows (Čiegis, Iliev, Starikovičius, Steiner 2006), and pattern formation in reaction-diffusion sys-

tems (see a fundamental book of Murray (Murray 2003) and references given in it). The chemotaxis-driven instability is investigated in many papers and different types of dynamic behaviour of nonlinear systems can be observed, see, e.g., the classic Keller-Segel model (Keller, Segel 1970; Murray 2003).

The dynamics of nonlinear systems can be investigated by various mathematical techniques, an extensive review with many examples and applications can be found e.g., in (Murray 2002; Strogatz 1994) and references contained in these books.

In our analysis a new approach is applied, when the chemotaxis-driven instability is correlated to the ill-posed problem defined by the backward in time diffusion process.

Let us assume that parameter γ is sufficiently large, but $\gamma r \sim \mathcal{O}(1)$, then due to the fast relaxation we get that

$$v \approx \frac{u^p}{1+\beta u^p}, \quad x \in (0,1), \;\; t > 0.$$

Then equation (3.1) can be written as a nonlinear diffusion-reaction equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - \frac{\partial}{\partial x}\left(\frac{\chi p u^p}{(1+\alpha v)^2(1+\beta u^p)^2}\frac{\partial u}{\partial x}\right) + \gamma r u(1-u). \qquad (3.4)$$

For some values of parameters equation (3.4) describes a backward in time parabolic problem and therefore it can be ill-posed (Engl, Hanke, Neubauer 1996; Tikhonov, Arsenin 1977)). In such a case, its solution does not depend continuously on initial data and small perturbations can lead to unbounded changes of the solution. Note that values of parameters, which were used in (Baronas, Šimkus 2011) for computational simulations, belong to this critical set.

Analysis of the backward parabolic equation. As an example a linear backward parabolic equation with the Dirichlet boundary conditions is considered:

$$\begin{aligned} &\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \quad t < T, \qquad (3.5)\\ &u(x,T) = u_0(x), \quad 0 \le x \le 1,\\ &u(0,t) = 0, \quad u(1,t) = 0. \end{aligned}$$

Let define the final condition $u_0(x)$ as a solution of the forward parabolic equation:

$$\begin{aligned} &\frac{\partial v}{\partial t} = D\frac{\partial^2 v}{\partial x^2}, \quad t > 0, \\ &v(x,0) = \exp(-(x-0.5)^6), \quad 0 \le x \le 1, \\ &v(0,t) = 0, \quad v(1,t) = 0. \end{aligned} \tag{3.6}$$

at $t = T$, i.e. $u_0(x) = v(x,T)$.

The well-known properties of the forward parabolic equation are straightforwardly obtained by applying the standard Fourier analysis. Let us assume that the initial condition can be written as $v(x,0) = \sum_{n=1}^{N} c_n \sin(\pi n x)$. Then a solution of parabolic problem (3.6) is given by

$$v(x,t) = \sum_{n=1}^{N} c_n e^{-D\pi^2 n^2 t} \sin(\pi n x).$$

It is shown, that high modes of the Fourier sum are damped faster and $v(x.t)$ becomes smoother for $t > 0$. Fig. 3.3 shows the initial condition $v(x,0)$ and the solution $v(x,T)$ of (3.6) at the final time $T = 0.1$. The diffusion coefficient $D = 0.1$ is used in all computations.

The Fourier stability analysis of the backward parabolic problem can be done in a similar way:

$$v(x,t) = \sum_{n=1}^{N} c_n e^{D\pi^2 n^2 (T-t)} \sin(\pi n x).$$

Now is is seen that high modes of the Fourier sum are growing faster and noise perturbations (e.g., corresponding to a white noise) are amplified for $t > 0$.

Fig. 3.4 shows a solution $u(x,0)$ of the backward parabolic problem at time moment $t = 0$ (the initial condition $u(x,T)$ is defined by the solution $v(x,T)$ of problem (3.6)).

Regularization of the backward parabolic equation can be done in various ways. Here mention two approaches are mentioned. In the first method the

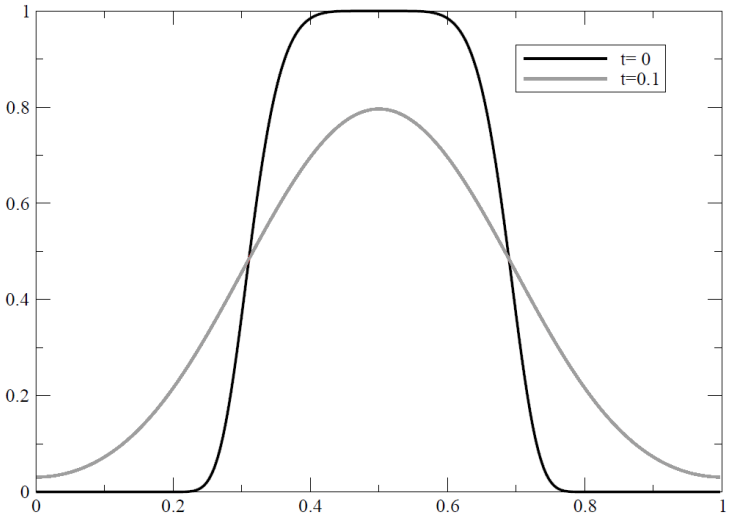


Fig. 3.3. Plots of the initial condition $v(x,0)$ and the solution $v(x,T)$ of problem (3.6)

regularization is done by a nonlocal boundary value problem (Nho Hao 2010):

$$\frac{\partial u_\alpha}{\partial t} = D\frac{\partial^2 u_\alpha}{\partial x^2}, \quad 0 < t < aT, \tag{3.7}$$
$$\alpha u_\alpha(x,0) + u_\alpha(x,aT) = u_0(x), \quad 0 \le x \le 1,$$

with $a > 1$ being given data and $\alpha > 0$, the regularization parameter. As an approximation of $u(x,t)$ take $u_\alpha\big((a-1)T+t\big)$. A priori and a posteriori rules for a selection of regularization parameter α are proposed in (Nho Hao 2010), which yield order optimal regularization methods. The standard Fourier analysis gives an explicit form of the solution

$$u_\alpha(t) = \sum_{n=1}^{N} c_n \frac{\exp\left(-\pi^2 n^2 Dt\right)}{\alpha + \exp\left(-\pi^2 n^2 DaT\right)} \sin(\pi n x),$$

from which the well-posedness of the regularized problem (3.7) follows.

In the second approach the ill-posed backward parabolic problem is regularized by a perturbed well-posed backward parabolic equation (Tikhonov,

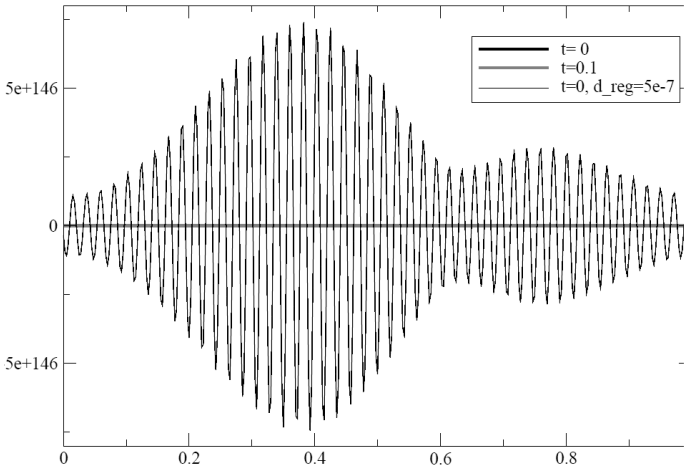


Fig. 3.4. Plots of solutions of the backward parabolic problem at $t = 0$ when the regularization parameter is too small

Arsenin 1977):

$$\begin{aligned} &\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + \alpha\frac{\partial^4 u}{\partial x^4}, \quad 0 < t < T, \\ &u(x,T) = u_0(x), \quad 0 \le x \le 1, \end{aligned} \tag{3.8}$$

with $\alpha > 0$ being the regularization parameter. The Fourier analysis explains the regularization effect:

$$u(x,t) = \sum_{n=1}^{N} c_n e^{\pi^2 n^2 (D - \alpha\,\pi^2 n^2)(T-t)} \sin(\pi n x).$$

It is shown that high modes of the Fourier sum are damped fast and therefore noise perturbations are not amplified for $t < T$. Fig. 3.5 shows a solution of the regularized problem (3.8).

Note, that due to non-linearity of the diffusion coefficient, the ill-posedness of the problem is self-limiting, since

$$\frac{\chi p u^p}{(1+\alpha v)^2(1+\beta u^p)^2} = \mathcal{O}(1/u^p) \quad \text{for } u^p \gg 1,$$

i.e. the ill-posedness occurs inside specific interval of $u < u_r$, where u_r

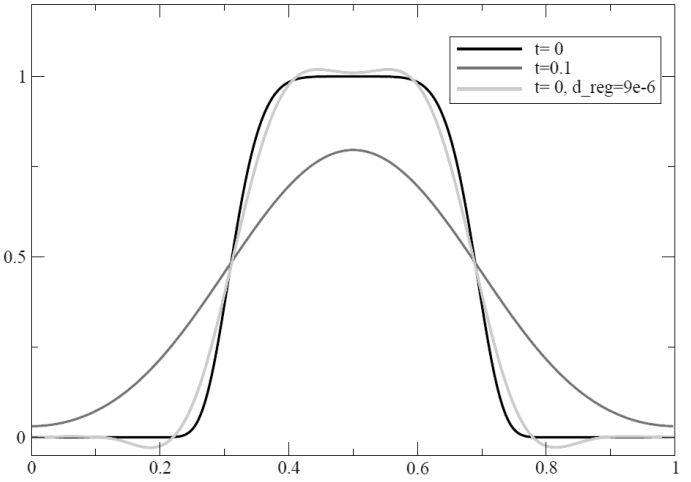


Fig. 3.5. Plots of solutions of the backward parabolic problem at $t = 0$ when the regularization parameter $\alpha = 0.9 \cdot 10^{-5}$

3.2. The Finite Difference Scheme

In this section present numerical techniques which are used to approximate solutions of system (3.1)–(3.3) are presented. A comprehensive treatment of theoretical and implementation issues of discretization methods for advection-diffusion-reaction problems is given in the monograph by Hundsdorfer, Verwer (2003). Very interesting applications of these results for biomedical problems are described in (Gerisch, Chaplan 2006), (Gerisch, Verwer 2002) (see also references given in these publications).

To obtain the discrete scheme the following techniques are applied:

1. The method of lines (MOL) (Gerisch, Chaplan 2006; Hundsdorfer, Verwer 2003) is used.

2. In order to speed-up calculations operator splitting method is used and advection process part using efficient explicit solvers is solved.

It is used a linearized implicit backward Euler method for the approximation of the diffusion-reaction subproblems and the explicit forward Euler method for solution of the advection subproblem. Lets restrict to the first order methods due to their robust stability. Note that it is important to investigate the influence of a possible ill-posedness of the PDE system to the asymptotical behaviour of the solution.

Domain $[0,1]$ is covered by a discrete uniform grid

$$\omega_h = \{x_j : x_j = jh,\ j = 0,\dots,N-1\},\ \ x_N = 1$$

with the grid points x_j. On the semidiscrete domain $\omega \times [0,T]$ lets define functions $U_j(t) = U(x_j,t)$, $V_j(t) = V(x_j,t)$, $j = 0,\dots,N-1$, here U_j, V_j are approximations of solutions $u(x_j,t)$, $v(x_j,t)$ on the discrete grid ω_h at time moment t. Lets also define the forward and backward space finite differences with respect to x:

$$\partial_x U_j = \frac{U_{j+1} - U_j}{h}, \quad \partial_{\bar{x}} U_j = \frac{U_j - U_{j-1}}{h}.$$

Operators:

$$A_T(U,V,j) = \frac{1}{h}\big(F_T(U,a,j+1/2) - F_T(U,a,j-1/2)\big). \tag{3.9}$$

$$A_{DR2}(V,U,j) = \partial_{\bar{x}}\partial_x V_j + \gamma\left(\frac{U_j^p}{1+\beta U_j^p} - V_j\right).$$

Then the split-type algorithm is defined by

$$\frac{U_j^{n+\frac{1}{3}} - U_j^n}{0.5\tau} = A_T(U^n, V^n, j), \tag{3.10}$$

$$\frac{U_j^{n+\frac{2}{3}} - U_j^{n+\frac{1}{3}}}{\tau} = D\partial_{\bar{x}}\partial_x U_j^{n+\frac{2}{3}} + \gamma r U_j^{n+\frac{1}{3}}\Big(1 - U_j^{n+\frac{2}{3}}\Big), \tag{3.11}$$

$$\frac{U_j^{n+1} - U_j^{n+\frac{2}{3}}}{0.5\tau} = A_T(U^{n+\frac{2}{3}}, V^n, j), \tag{3.12}$$

$$\frac{V_j^{n+1} - V_j^n}{\tau} = A_{DR2}(V^{n+1}, U^{n+1}, j). \tag{3.13}$$

$U_j^{n+a} = U(x_j, t_{n+a\tau})$, $V_j(t) = V(x_j,t)$, $t_{j+1} = t_j + \tau$ $j = 0,\dots,N-1$. So by applying (3.10–3.13) to U_j we get U_{j+1}. Steps (3.10) and (3.12) scheme parts are explicit (calculated by applying the formula), (3.11) and (3.13) are implicit (calculated by solving system of equations). The details of obtaining scheme (3.10) are provided in Appendix B.

3.3. Numerical Experiments

In this section, the results of numerical experiments in order to verify our theoretical investigations are presented.

3.3.1. Convergence Analysis

The first goal is to investigate the convergence of the finite difference discretizations of the PDE model (3.1) in the case when dynamics of solutions leads to formation of complicated spatial-temporal patterns. The development of the bacterial population was simulated for the following values of the model parameters (Baronas, Šimkus 2011):

$$D = 0.1, \;\; \chi = 9.2, \;\; r = 1, \;\; \alpha = 0.7, \;\; \beta = 1.4, \;\; \gamma = 625, \;\; p = 2.$$

First, consider the MOL type discrete scheme (3.10)–(3.13), when the space grid step has been fixed to $h = 0.005$, and computations have been performed with different time steps $\tau = 1 \cdot 10^{-6}, 5 \cdot 10^{-7}, 2.5 \cdot 10^{-7}$. A simple initial condition

$$u_0(x) = 1 + 0.2 \sin(4\pi x), \quad x \in [0, 1] \tag{3.14}$$

is used in computations. Fig. 3.6 shows the profiles of function $U_j(t^n)$ at time $T_f = 0.6$.

As can be seen, the discrete solutions do not converge in the classical norm when the discretization temporal step τ is decreased.

A similar behaviour of numerical approximations is also observed for the fully discrete scheme (3.10)–(3.13). Fig. 3.7 shows the results, when the spatial and temporal grid steps are connected by the relation $\tau = 5 \cdot 10^{-5} h$ and $h = 0.01, 0.005, 0.0025$.

In order to prove that such convergence behaviour of discrete solutions depends on dynamical instability of the mathematical model (3.1) and more exactly it is a consequence of negative diffusion driven instability for a backward in time parabolic problem, consider the same problem (3.1) but for a short time interval $t \in (0, T_f]$. During this time interval the diffusion coefficient D still dominates the chemotaxis perturbations and the mathematical model defines a well–posed problem. Dynamics of the maximal advection velocity $v_{max}(t)$ is presented in Fig. 3.8, thus it can be taken $T_f = 0.125$.

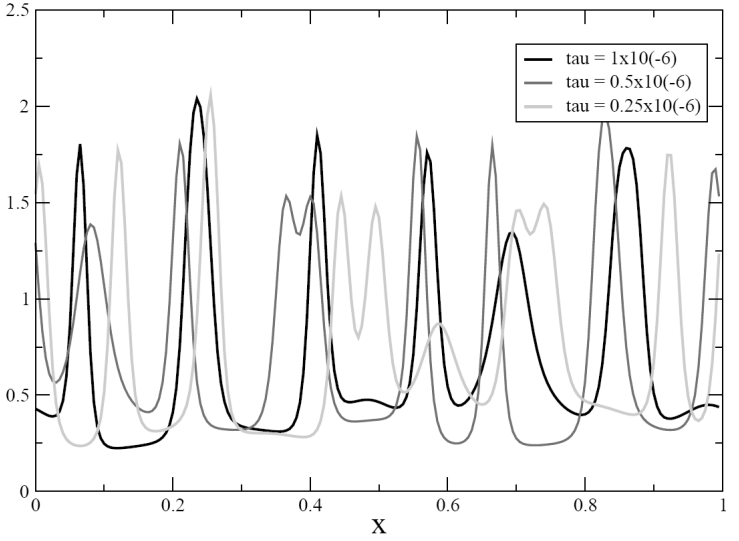


Fig. 3.6. The profiles of function $U_j(t^n)$ at time $T_f = 0.6$ for fixed $h = 0.005$ and different time steps

Fig. 3.9 shows the profiles of function $U_j(t^n)$ at time $T_f = 0.125$. Computations are done with $h = 0.005$ and different values of time step. The discrete solutions converge as expected according to the standard error estimates.

Next, it is investigated a sensitivity of the numerical solution with respect the initial conditions computed at some time moment, when the solution have reached a phase of patern formation. The main goal is to determine a length of time interval for which a solution still remembers the influence of the given initial conditions. In our experiments we have computed a solution of (3.10)–(3.13) till $t = 1.5$ with $h = 1/4000$ and used it as initial conditions for the following computations.

From Table 3.1 it is seen that lowering τ has no significant effect on convergence interval $[1.5, T]$

3.3.2. Formation of Complex Patterns

In order to investigate the dependence of pattern formation on the initial condition, the space-time plots $U(x, t)$ was simulated for two types of initial con-

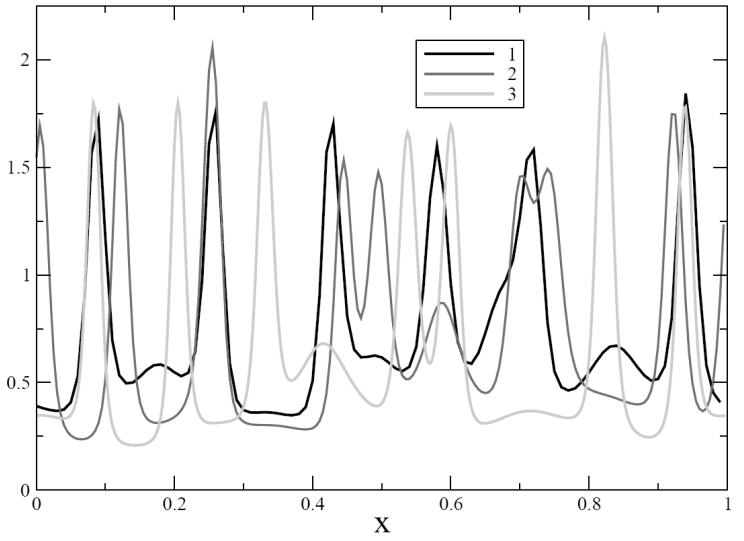


Fig. 3.7. The profiles of function $U_j(t^n)$ at time $T_f = 0.6$ for $\tau = 5 \cdot 10^{-5} h$ and different space grid steps

Table 3.1. Sensitivity analysis of the of the solution of mathematical model with respect the initial conditions. Errors of the discrete solution of (3.10)–(3.13), when a discrete solution at $t = 1.5$ with $h = 1/4000$ is used as the initial condition

num. of nodes	time step	$T = 1.52$	$T = 1.54$	$T = 1.58$	$T = 1.65$
$J = 250$	$\tau = 2 \cdot 10^{-6}$	0.423	1.920	2.398	2.585
$J = 500$	$\tau = 1 \cdot 10^{-6}$	0.092	0.340	1.159	2.942
$J = 1000$	$\tau = 5 \cdot 10^{-7}$	0.030	0.083	0.324	2.146
$J = 2000$	$\tau = 2.5 \cdot 10^{-7}$	0.008	0.019	0.108	1.687

ditions:

$$u(x, 0) = 1 + 0.2 \sin(2\pi x) \quad \text{and} \quad u(x, 0) = 1 + 0.2 \sin(4\pi x).$$

The results of simulations are depicted in Fig. 3.10.

As it is seen, both patterns simulated for different initial conditions are similar in a qualitative sense.

Finally, it was added the regularization term into the first equation of system (3.1). The simulations show that if the regularization parameter α is selected such, that the perturbed problem starts to be well-posed, the pattern formation disappears and the cell density converges to a uniform stable stationary state.

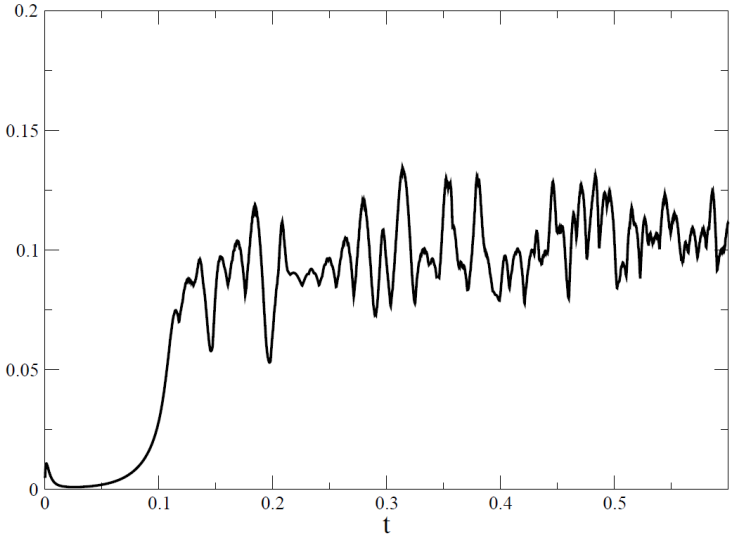


Fig. 3.8. Dynamics of the maximal advection velocity $v_{max}(t)$

3.4. Parallel Numerical Algorithms

In this section algorithms for parallel computing of considered problem is proposed, the efficiency and scalability of parallel algorithms are investigated.

2D Generalization The model (3.1) is one dimensional, hence often it is not reasonable to speedup calculations with parallel computing in that case. Therefore lets generalize that model extending it to the two dimensional model

$$\frac{\partial u}{\partial t} = \sum_{j=1}^{2} \left[D \frac{\partial^2 u}{\partial x_j^2} - \frac{\partial}{\partial x_j} \left(\frac{\chi u}{(1+\alpha v)^2} \frac{\partial v}{\partial x_j} \right) \right] + \gamma r u (1-u), \qquad (3.15)$$

$$\frac{\partial v}{\partial t} = \sum_{j=1}^{2} \frac{\partial^2 v}{\partial x_j^2} + \gamma \Big(\frac{u^p}{1+\beta u^p} - v \Big), \quad X \in (0,1) \times (0,1), \quad t > 0,$$

where u is the dimensionless cell density, v is the dimensionless chemoattractant concentration, D defines the constant cell diffusion, χ is the chemotactic coefficient, r denotes the growth rate, α defines the receptor sensitivity, β stand for saturating of the signal production, γ defines the ratio of the spatial and temporal scales, and $p \geq 1$. Boundary conditions can vary according to the

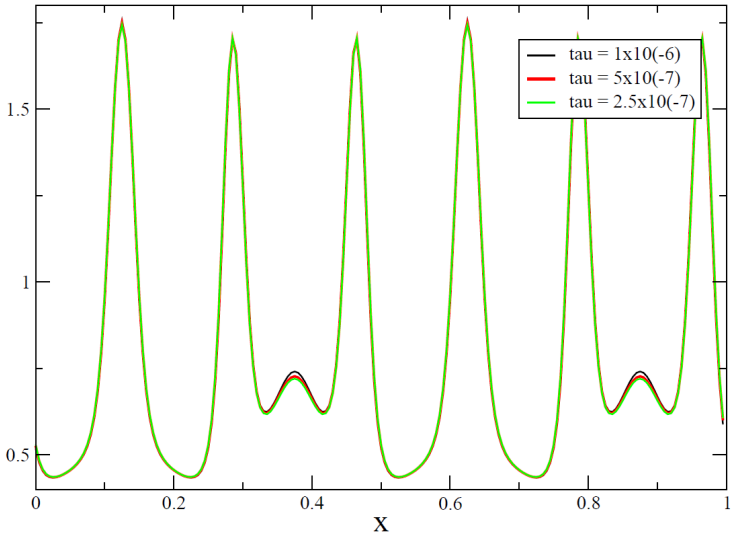
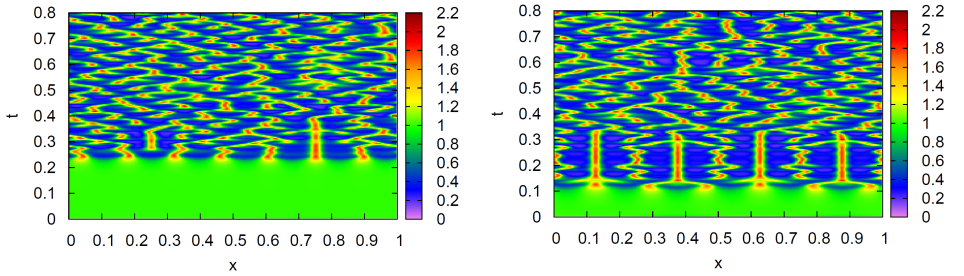


Fig. 3.9. The profiles of function $U_j(t^n)$ at time $T_f = 0.125$ for fixed $h = 0.005$ and different time steps



a) $u(x,0) = 1 + 0.2\sin(2\pi x)$, b) $u(x,0) = 1 + 0.2\sin(4\pi x)$

Fig. 3.10. Space-time plots of the cell density $U(x,t)$ simulated for different initial conditions

biological system, zero-flux and periodical conditions are popular and natural choices. Here, the periodicity conditions were used.

To approximate the two dimensional model the same approximation technology was used as was used in the one dimensional case.

The method of lines. Discretization in space. At the first step the spatial derivatives are approximated in the PDEs (3.15) by applying robust and accurate approximations targeted for special physical processes described by differential equations. The computational domain $\omega := (0,1)^2$ is covered

by an equi-spaced grid ω_h with J computational cells in each direction, the width of the cell is denoted by $h := 1/J$. On the semidiscrete domain $\omega_h \times [0,T]$ lets define functions $U_{ij}(t) = U(x_{1i}, x_{2j}, t)$, $V_{ij}(t) = V(x_{1i}, x_{2j}, t)$, $ij = 0, \ldots, N-1$, here U_{ij}, V_{ij} approximate exact solutions $u(x_1, x_2, t)$, $v(x_1, x_2, t)$ on the discrete grid ω_h at time moment t.

Using the finite volume approach, lets approximate the diffusion and reaction terms by the following finite difference equations:

$$A_{D_k}(U) = D\partial_{\bar{x}_k}\partial_{x_k}U_{ij}, \;\; k = 1,2, \; A_R U = \gamma r U_{ij}(1 - U_{ij}), \; (x_{1i}, x_{2j}) \in \omega_h,$$
$$\widetilde{A}_{D_1}(V) = \partial_{\bar{x}_1}\partial_{x_1}V_{ij} - \gamma V_{ij}, \quad \widetilde{A}_{D_2}(V) = \partial_{\bar{x}_2}\partial_{x_2}V_{ij}$$
$$\widetilde{A}_R U = \gamma \frac{U_{ij}^p}{1 + \beta U_{ij}^p}, \quad (x_{1i}, x_{2j}) \in \omega_h.$$

For chemotaxis term lets consider the upwind-based discrete 1D fluxes, e.g., for x_2 coordinate (Čiegis, Bugajev 2012; Hundsdorfer, Verwer 2003):

$$F_T(U, a, 2) = a_{i,j+\frac{1}{2}}\big[U_{ij} + \psi(\theta_{ij})\big(U_{i,j+1} - U_{ij}\big)\big], \quad a_{i,j+\frac{1}{2}} \geq 0,$$
$$F_T(U, a, 2, j + \frac{1}{2}) = a_{i,j+\frac{1}{2}}\big[U_{i,j+1} + \psi(1/\theta_{i,j+1})\big(U_{ij} - U_{i,j+1}\big)\big], a_{i,j+\frac{1}{2}} < 0,$$

with the Koren limiter function. Here the discrete spatial approximation of the velocity is computed by

$$a_{i,j+\frac{1}{2}}(t) = -\frac{\chi}{\big(1 + \alpha(V_{ij} + V_{i,j+1})/2\big)^2}\,\partial_{x_2}V_{ij}.$$

Lets denote the discrete advection operator as

$$A_T(U,V) = \frac{1}{h}\big(F_T(U, a, 1, i + 1/2) - F_T(U, a, 1, i - 1/2)\big) + \frac{1}{h}\big(F_T(U, a, 2, j + 1/2) - F_T(U, a, 2, j - 1/2)\big).$$

Then it is obtained a nonlinear ODE system for the evolution of semidiscrete solutions

$$\frac{dU}{dt} = A_T(U,V) + A_{D_1}(U) + A_{D_2}(U) + A_R U, \tag{3.16}$$
$$\frac{dV}{dt} = \widetilde{A}_{D_1}(V) + \widetilde{A}_{D_2}(V) + \widetilde{A}_R(U),$$

here discrete periodical boundary conditions are included into the definition of discrete diffusion and advection operators.

Time stepping. In order to develop efficient solvers in time for the obtained large ODE systems the different nature of the discrete operators defining the advection and the diffusion-reaction processes are taken into account.

Let ω_τ be a uniform temporal grid

$$\omega_\tau = \{t^n : t^n = n\tau,\, n = 0, \dots, M,\, M\tau = T_f\},$$

here τ is the time step.

Given approximations U_j^n, V_j^n at time t^n, compute solutions at $t^{n+1} = t^n + \tau$ by the following IMEX ADI type scheme (see introduction to this topic by Hundsdorfer (2000)): A_T is a non-stiff term suitable for explicit time integration

$$\widetilde{U}_{ij}^n = U_{ij}^n + \tau A_T(U^n, V^n), \quad (x_{1i}, x_{2j}) \in \omega_h, \tag{3.17}$$

A_{D_k}, $\widetilde{A}_{D_k}$, A_R, $\widetilde{A}_R$ are stiff terms requiring an implicit ADI treatment

$$U_{ij}^{n+\frac{1}{2}} = \widetilde{U}_{ij}^n + \tau\big(A_{D_1}(U^{n+\frac{1}{2}}) + A_{D_2}(U^n) + A_R(U^{n+\frac{1}{2}})\big), \tag{3.18}$$

$$U_{ij}^{n+1} = U_{ij}^{n+\frac{1}{2}} + \tau\big(A_{D_2}(U^{n+1}) - A_{D_2}(U^n)\big),$$

$$V_{ij}^{n+\frac{1}{2}} = V_{ij}^n + \tau\big(\widetilde{A}_{D_1}(V^{n+\frac{1}{2}}) + \widetilde{A}_{D_2}(V^n) + \widetilde{A}_R(U^{n+1})\big), \tag{3.19}$$

$$V_{ij}^{n+1} = V_{ij}^{n+\frac{1}{2}} + \tau\big(\widetilde{A}_{D_2}(V^{n+1}) - \widetilde{A}_{D_2}(V^n)\big).$$

Such a mixture of implicit and explicit methods gives efficient solvers for each sub-step of the algorithm and due to the full approximation the stationary solution exactly satisfies the finite difference scheme.

The accuracy of the approximation in time can be increased by applying the symmetrical version of this scheme, for details see (Hundsdorfer, Verwer 2003).

3.5. Parallel Algorithm

Calculations (3.18)–(3.19) can take a lot of computer resources to solve this direct problem, moreover, it can be a part of optimization problem, when it

is needed to solve the two dimensional problem many times (1000 times or more). So the speed of calculations can be an issue and in this case the development of parallel version of algorithm become important. The parallel version of algorithm is presented in this section.

In order to evaluate the parallel algorithms it is necessary to investigate the efficiency and scalability of parallel numerical algorithms for approximation of problem (3.15). Thus in order to find a bound on the parallel efficiency of split type approximations and to measure the influence of different parts of the full mathematical model, we have increased the amount of computations which can be done in parallel by modifying the basic full-approximation scheme (3.17)–(3.19). Using the operator splitting method, we further split operator $A_{D_1} + A_R$ by separating diffusion and nonlinear reaction terms. Note that for the obtained new discrete scheme the requirement of full approximation of the stationary solution is not satisfied.

$$U_{ij}{}^{n+\frac{1}{4}} = U_{ij}^{n} + \tau A_T(U^n, V^n), \quad (x_{1i}, x_{2j}) \in \omega_h, \tag{3.20}$$

$$U_{ij}{}^{n+\frac{m+s}{4m}} = U_{ij}{}^{n+\frac{m+s-1}{4m}} + \frac{\tau}{m}\gamma r U_{ij}^{n+\frac{m+s-1}{4m}}\big(1-U_{ij}^{n+\frac{m+s}{4m}}\big), \quad s = 1, \ldots, m, \tag{3.21}$$

$$U_{ij}^{n+\frac{3}{4}} = U_{ij}^{n+\frac{1}{2}} + \tau\big(A_{D_1}(U^{n+\frac{3}{4}}) + A_{D_2}(U^{n+\frac{1}{2}})\big), \tag{3.22}$$

$$U_{ij}^{n+1} = U_{ij}^{n+\frac{3}{4}} + \tau\big(A_{D_2}(U^{n+1}) - A_{D_2}(U^{n+\frac{1}{2}})\big),$$

$$V_{ij}^{n+\frac{1}{2}} = V_{ij}^{n} + \tau\big(\widetilde{A}_{D_1}(V^{n+\frac{1}{2}}) + \widetilde{A}_{D_2}(V^n) + \widetilde{A}_R(U^{n+1})\big), \tag{3.23}$$

$$V_{ij}^{n+1} = V_{ij}^{n+\frac{1}{2}} + \tau\big(\widetilde{A}_{D_2}(V^{n+1}) - \widetilde{A}_{D_2}(V^n)\big).$$

The reaction problem (3.21) is split into $m \geq 1$ sub-steps in order to resolve a stiff reaction problem accurately, we used here the possibility to model separate processes with different time steps. Note that such an additional split step does not change the asymptotical efficiency of the parallel algorithm, as it is shown by results of the scalability analysis.

A parallel version of the proposed algorithm is developed by using the domain decomposition method. The two-dimensional decomposition of the space mesh ω_h into $P = P_1 \times P_2$ subdomains is applied. The advection and reaction steps are resolved by the same sequential algorithms, only data ex-

change among processors is implemented when the stencil of the finite difference scheme leads to overlapping of local sub-meshes of different processors.

The standard factorization algorithm for solving linear systems of equations with tridiagonal matrices is fully sequential and should be changed to some parallel solver. Here use the well-known Wang's algorithm (Wang 1981)) is used, but the modification is done in order to adapt this algorithm for periodic boundary conditions. Similar questions are also investigated for parabolic and psoudoparabolic problems with periodic and nonlocal boundary conditions in our article "Parallel algorithms for three-dimensional parabolic and pseudoparabolic problems with different boundary conditions".

3.5.1. Parallel Factorization Algorithm

Let denote by U_j the 1D vector of unknowns, corresponding to 2D vector $\{U_{ij}\}$, where one coordinate, e.g. $1 \le j \le J$, is fixed:

$$U_j = \big(U_{1j}, U_{2j}, \ldots, U_{M_1,j}\big),$$

where $M_1 = J/P_1$ is the number of unknowns in x_1 direction belonging to each processor. Thus we solve M_1 one dimensional systems with tridiagonal matrix, where periodic boundary conditions are taken into account in the first and the last equations:

$$\begin{cases} -a_1 U_J + c_1 U_1 - b_1 U_2 = f_1, \\ -a_j U_{j-1} + c_j U_j - b_j U_{j+1} = f_j, \quad j = 2, \ldots, J-1, \\ -a_J U_{J-1} + c_J U_J - b_J U_1 = f_J. \end{cases} \tag{3.24}$$

Let assume that P_2 processes are used to solve this system. Lets partition the system into P_2 blocks and denote the number of unknowns in each block by $M_2 = J/P_2$. The p-th process updates a block of equations for $j = s_p, \ldots, f_p$, where $s_p = (p-1)M_2 + 1$, $f_p = pM_2$.

1. First, using the modified forward factorization algorithm, vector U_j is expressed in the following form

$$\begin{gathered} U_j = \alpha_j U_{j+1} + \gamma_j U_{s_p - 1} + \beta_j, \\ j = s_p, \ldots, f_p, \ s_1 - 1 := J, \ f_P + 1 := 1, \end{gathered} \tag{3.25}$$

where

$$\alpha_{s_p} = \frac{b_{s_p}}{c_{s_p}}, \quad \gamma_{s_p} = \frac{a_{s_p}}{c_{s_p}}, \quad \beta_{s_p} = \frac{F_{s_p}}{c_{s_p}},$$
$$\alpha_j = \frac{b_j}{c_j - a_j\alpha_{j-1}}, \quad \gamma_j = \frac{a_j\gamma_{j-1}}{c_j - a_j\alpha_{j-1}}, \quad \beta_j = \frac{F_j + a_j\beta_{j-1}}{c_j - a_j\alpha_{j-1}}.$$

All processes implement the first step in parallel and the complexity of it is $8M_1M_2$ flops. No data communication among processors is required.

2. Second, the block matrix is diagonalized, except the first and last columns of the block, i.e. vector U_j is expressed in the form

$$U_j = \alpha_j U_{f_p} + \gamma_j U_{s_p-1} + \beta_j, \quad j = s_p, \ldots, f_p, \tag{3.26}$$

where factorization coefficients are recomputed as:

$$\alpha_j = \alpha_{j+1}\alpha_j, \ \gamma_j = \gamma_j + \gamma_{j+1}\alpha_j, \ \beta_j = \beta_j + \beta_{j+1}\alpha_j, \quad j = f_p - 1, \ldots, s_p.$$

The computations are implemented from right to left. All processes run the second step in parallel and the complexity of it is $5M_1M_2$ flops. No data communication among processors is required.

3. In the third step, all processes $p > 1$, excluding the first one, send their first row of the matrix to its neighbour $(p-1)$, then all processes $p < P_2$, excluding the last one, modify the last row:

$$U_{f_p} = \alpha_{f_p} U_{f_{p+1}} + \gamma_{j_p} U_{f_{p-1}} + \beta_{f_p}, \quad p = 1, \ldots, P_2 - 1. \tag{3.27}$$

where

$$\alpha_{f_p} = \frac{\alpha_{f_{p+1}}\alpha_{f_p}}{1 - \gamma_{f_{p+1}}\alpha_{f_p}}, \quad \gamma_{f_p} = \frac{\gamma_{f_p}}{1 - \gamma_{f_{p+1}}\alpha_{f_p}}, \quad \beta_{f_p} = \frac{\beta_{f_p} + \beta_{f_{p+1}}\alpha_{f_p}}{1 - \gamma_{f_{p+1}}\alpha_{f_p}}.$$

All processes implement the third step in parallel and the complexity of it is $8M_1$ flops. They exchange between neighbours $3M_1$ elements. The complexity of data communication is $2\big(\alpha + 3\beta M_1\big)$, where α is the message startup time, β is the time required to send one element of data.

Now, taking the last equation of each local sub-system and adding to the new system the first equation of the first process gives $(P_2+1)\times(P_2+1)$ tridiagonal system of linear vector equations

$$\begin{cases} U_1 = \alpha_1 U_{f_1} + \gamma_1 U_J + \beta_1, \\ U_{f_p} = \alpha_{f_p} U_{f_{p+1}} + \gamma_{f_p} U_{f_{p-1}} + \beta_{f_p}, \quad p = 1, \ldots, P_2 - 1. \\ U_J = \alpha_J U_1 + \gamma_J U_{f_{p-1}} + \beta_J. \end{cases} \tag{3.28}$$

The obtained system of linear equations again depends on periodic boundary conditions. It can be solved sequentially by one process and results distributed to the remaining processes. Lets propose to use a two-side factorization algorithm: all processes are implementing the factorization algorithm, but local subtasks are computed sequentially, the required data is exchanged among neighbour processes only. That lets to avoid any global data reduction and distribution operations. A small scale parallelization of the algorithm is still preserved since the computations are divided among two groups of processes.

4. In the fourth step, processes are divided into two groups and sequentially sweep from left to the right transformations of the last row of local system. E.g., process $p > 1$ of the left group gets $(p-1)$-th neighbour's last equation, then modifies its own last equation and sends new coefficients to $(p+1)$-th neighbour. The last equation of the local subsystem is transformed to the form:

$$\begin{aligned} U_{f_p} &= \alpha_{f_p} U_{f_{p+1}} + \gamma_{f_p} U_J + \beta_{f_p}, \quad p = 2, \ldots, P_2/2, \\ U_{f_p} &= \gamma_{f_p} U_{f_{p-1}} + \alpha_{f_p} U_J + \beta_{f_p}, \quad p = P_2/2 + 1, \ldots, P_2 - 1, \end{aligned} \tag{3.29}$$

where, e.g. for $p \le P_2/2$:

$$\alpha_{f_p} = \frac{\alpha_{f_p}}{1 - \gamma_{f_p}\alpha_{f_{p-1}}}, \quad \gamma_{f_p} = \frac{\gamma_{f_p}\gamma_{f_{p-1}}}{1 - \gamma_{f_p}\alpha_{f_{p-1}}},$$
$$\beta_{f_p} = \frac{\beta_{f_p} + \beta_{f_{p-1}}\gamma_{f_p}}{1 - \gamma_{f_p}\alpha_{f_{p-1}}}.$$

The computational complexity of the fourth step is $4P_2M_1$ flops and the complexity of data communication is $\big(\alpha + 3\beta M_1\big)P_2$.

5. In the fifth step, both groups exchange information, and then sequen-

tially transform the last raw equations into the form:

$$U_{f_p} = \gamma_{f_p} U_J + \beta_{f_p}, \quad p = P_2/2, \ldots, 1, \quad U_1 = \gamma_1 U_J + \beta_1,$$

$$U_{f_p} = \alpha_{f_p} U_J + \beta_{f_p}, \quad p = P_2/2 + 1, \ldots, P_2 - 1, \tag{3.30}$$
$$U_J = \alpha_J U_1 + \beta_J,$$

where, e.g. for $p \le P_2/2$:

$$\gamma_{f_p} = \gamma_{f_p} + \alpha_{f_p} \gamma_{f_{p+1}}, \quad \beta_{f_p} = \beta_{f_p} + \beta_{f_{p+1}} \alpha_{f_p}.$$

The computational complexity of the fifth step is $2P_2M_1$ flops and the complexity of data communication is $\big(\alpha + 2\beta M_1\big)P_2$.

6. In the sixth step, the first and last processes exchange information on the first and last equations of the system, compute vectors U_1 and U_J and then all processes of the left and right groups compute sequentially the remaining vectors U_{f_p}. The computational complexity of the sixth step is P_2M_1 flops and the complexity of data communication is $\big(\alpha + \beta M_1\big)P_2$.
7. In the last step, all processes in parallel compute vectors U_j, $j = 1, \ldots, J$. The computational complexity of this step is $4M_1M_2$ flops.

The same algorithm for solution of linear systems of the ADI scheme resolving 1D diffusion process A_{D_1} is applied. Thus the total complexity of the parallel factorization algorithm to solve a 2D diffusion problem with periodical boundary conditions is equal to

$$T_{D,P} = 34\frac{J^2}{P} + 7\Big(\frac{P_1}{P_2} + \frac{P_2}{P_1}\Big)J + 8\Big(\frac{1}{P_1} + \frac{1}{P_2}\Big)J + \alpha\big(2 + 3(P_1 + P_2)\big) + 3\beta\Big(1 + 2\Big(\frac{P_1}{P_2} + \frac{P_2}{P_1}\Big)\Big)J. \tag{3.31}$$

Note that the complexity of the sequential factorization algorithm is $28J^2$ floating point operations, thus the additional costs of the parallel algorithm mainly depend on data communication costs.

In the case of the Dirichlet or Neumann boundary conditions the complexity of the sequential factorization algorithm is $16J^2$ operations, and the com-

plexity of the parallel algorithm implemented on one processor is $T_{D,1} = 34J^2$ operations, thus computational cost is increased more than twice.

3.5.2. Scalability Analysis

In this section the main scalability analysis results are given.

Nonlinear reaction step. Nonlinear reaction discrete equations (3.21) are solved by fully data parallel algorithm and no communication costs are required. The complexity of this part of operator splitting algorithm is

$$T_{R,P} = C_R \frac{J^2}{P}, \tag{3.32}$$

constant C_R depends on the number of splitting steps m, required to solve the stiff reaction equations in an accurate way.

Transport step. The transport step (3.20) parallel algorithm is data parallel and coincides with the sequential explicit algorithm. Additional communication costs depend on the stencil of the finite difference scheme. For the approximation of the chemotaxis advection process, values of function U on the cross stencil with one point radius are used, and values of function V on the cross stencil with two point radius are required. Thus the total complexity of the advection step is given by

$$T_{A,P} = C_T \frac{J^2}{P} + 4\alpha + 12\beta\Big(\frac{J}{P_1} + \frac{J}{P_2}\Big). \tag{3.33}$$

Adding all estimates (3.31)–(3.33), gives that the complexity of the parallel ADI algorithm (3.17)–(3.23) is given by

$$\begin{aligned} T_P = {} & (C_R + C_T + 68)\frac{J^2}{P} + 14\Big(\frac{P_2}{P_1} + \frac{P_1}{P_2}\Big)J + 16\Big(\frac{J}{P_1} + \frac{J}{P_2}\Big) \\ & + 2\alpha\big(4 + 3(P_1 + P_2)\big) + 6\beta\Big[1 + 2\Big(\frac{P_1}{P_2} + \frac{P_2}{P_1} + \frac{1}{P_1} + \frac{1}{P_2}\Big)\Big]J. \end{aligned}$$

Complexity of the sequential ADI algorithm (3.17)–(3.23) is given by

$$T_0 = (C_R + C_T + 56)J^2.$$

It follows from these estimates that the efficiency of the parallel algorithm is limited by the solution of reduced tridiagonal systems (3.28). Let us assume that $P_1 = P_2 = \sqrt{P}$, then the asymptotical optimal number of processors is determined from the equation, which defines the equidistribution of computational and data communication costs (the worst term of communication cost is of order cJ for $P_1 = P_2 = \sqrt{P}$):

$$\frac{J^2}{P} = cJ \quad \Longrightarrow \quad P = \mathcal{O}(J).$$

In future work, it is planned to solve diffusion sub-problems by using parallel iterative solvers based on AMG preconditioners, see e.g. (Evans, Žurič, Basara, Frolov 2012).

3.6. Computational Experiments

Computations were performed on Vilkas cluster of computers at Vilnius Gediminas Technical University, consisting of nodes with Intel®Core™ processor i7-860 @ 2.80 GHz and 4 GB DDR3-1600 RAM. Each of the four cores can complete up to four full instructions simultaneously. Results of computational experiments are given in Table 3.2.

Table 3.2. Scalability analysis of the parallel algorithm. The speed-up S_p and efficiency E_p coefficients for two problems of dimension 600×600 and 1200×1200

	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
$S_p(600)$	1.39	3.22	5.96	10.1	12.5
$E_p(600)$	0.70	0.81	0.75	0.63	0.39
$S_p(1200)$	1.32	2.60	5.58	12.0	20.9
$E_p(1200)$	0.66	0.65	0.70	0.75	0.65

It follows from the presented results, that parallel ADI type algorithms are efficient in solving real-world biochemistry applications. The modified Wang's algorithms efficiently solves systems of linear equations with tridiagonal matrices and periodic boundary conditions. Also it is seen that efficiency E_p do no decrease monotonically as p increases. It can be described by known CPU cache effect: each core has its own cache for computations, with more cores there are more cache in total, the bigger amount of cache is known to speed up the calculations. Other interpretation would be that smaller blocks of

data are easier to proceed, by saying easier we mean that the times of computations are not proportional to the size of data, but the times are smaller with small sizes of data. The conclusions of the scalability analysis about a linear scalability of the parallel algorithm are confirmed in computational experiments. In the future work it is planned to consider an alternative 1D domain decomposition, when the data is transposed after the solution of the tridiagonal systems of one direction, such that tridiagonal systems are always solved sequentially, as is done in 2D FFT codes. It is also planned to compare the direct ADI solvers with parallel solvers based on Krylov type iterative algorithms.

3.7. Summary of the Third Chapter

In this chapter one-dimensional diffusion-chemotaxis-reaction mathematical model was studied and have shown that chemotaxis-driven instability can be connected to the ill-posed problem defined by the backward in time diffusion process.

By using the well-established techniques the mathematical model is approximated by the discrete computational model. It is proved that the discrete solution inherits main properties of the solution of the differential mathematical model. Results of numerical experiments show that for such problems, where chemotaxis-driven instability defines the dynamics of a solution, the classical convergence property of numerical algorithms is not applicable. Instead of pointwise and similar convergence metrics it should be used qualitative criteria or averaged statistical characteristics (as, e.g., in the discrete element method).

The one-dimensional model can be too simple in order to obtain a satisfactory agreement with pattern formation seen in experiments. Thus 2D and 3D generalizations of the given mathematical model should be investigated. Construction of robust and efficient parallel algorithms can be done by using domain decomposition and splitting in space methods (Tumanova 2012).

3.8. Conclusions of the Third Chapter

1. The discrete solution inherits main properties of the solution of the differential mathematical model.
2. During the modelling of bacteria self-organization it was shown that chemotaxis-driven instability is connected to the ill-posed problem defined by the backward in time diffusion process.
3. Results of numerical experiments show that for such problems, where chemotaxis-driven instability defines the dynamics of a solution, the classical convergence property of numerical algorithms is not applicable. Instead of pointwise and similar convergence metrics we should use qualitative criteria or averaged statistical characteristics.
4. The one-dimensional model can be too simple in order to obtain a satisfactory agreement with pattern formation seen in experiments. Thus 2D and 3D generalizations of the given mathematical model should be investigated.

General Conclusions

1. The proposed model for prediction of computations time, which is based on detailed computations analysis, lets to evaluate different GPUs in terms of their efficiency for 3D FDTD calculations. However, the model should be modified to be more general so it could be applied to a newer GPUs.
2. The bottleneck of calculations is usually described by GPU bandwidth parameter. Additionally there is the possibility of efficient usage of L1 cache memory (this demonstrated in computational experiments).
3. Due to a problem specificity there are efficiency issues (the order of convergence is equal to 1) related to geometry and material parameters, when the OpenFOAM software was used – the usage of rectangular meshes and non-orthogonality corrections methods built-in into OpenFOAM decrease the FVM rate of solution convergence.
4. Domain triangulation with cell centres in Voronoi points eliminates non-orthogonality errors, keeping the FVM order of convergence (in order to obtain this result original OpenFOAM code was modified). The proposed technique solves the geometry related efficiency issues.

5. For solving real problem spatial adaptive mesh is essential and adaptive time steps also may be needed (heuristic for generating adaptive spatial mesh is proposed).
6. Harmonic mean formulas deal with big coefficients jumps, keeping the error small enough for modelling purposes (the usage of these formulas are controlled by a built-in option in OpenFOAM software).
7. The proposed special algorithm to control mesh singularities controls mesh singularities when modelling some (when the cables touch each other) real life cases.
8. The developed OpenFOAM solver to solve the the problem using distributed calculations shown that the scaling and efficiency performance on Vilkas cluster is good up to 32 cores when I/O effects are neglected and load balancing is used for mesh partition.
9. Smaller sizes of distributed discrete sub-problems enable a better caching and give a sub-linear speed-up (in considered case up to 30%) for computational part of the parallel algorithm.
10. It is important to test the effects of I/O costs when balancing between computation and I/O parts of the algorithm is not good, for example when a solution should be saved every 5-10 time steps.
11. The discrete solution inherits main properties of the solution of the differential mathematical model.
12. During the modelling of bacteria self-organization it was shown that chemotaxis-driven instability is connected to the ill-posed problem defined by the backward in time diffusion process.
13. Results of numerical experiments show that for such problems, where chemotaxis-driven instability defines the dynamics of a solution, the classical convergence property of numerical algorithms is not applicable. Instead of pointwise and similar convergence metrics we should use qualitative criteria or averaged statistical characteristics.
14. The one-dimensional model can be too simple in order to obtain a satisfactory agreement with pattern formation seen in experiments. Thus 2D and 3D generalizations of the given mathematical model should be investigated.

References

Adams, S.; Payne, J.; Boppana, R. Finite Difference Time Domain (FDTD) simulations using graphics processors. *DoD High Performance Computing Modernization Program Users Group Conference*, 334–338, 2007.

AlOnazi, A. Design and Optimization of OpenFOAM-based CFD Applications for Modern Hybrid and Heterogeneous HPC Platforms. *Master thesis, University College Dublin*, 2013.

Amiraliyev, G.; Cakir, M. Numerical solution of singularly perturbed problem with nonlocal boundary condition. *Appl. Math. Mech.*, 23(7): 755–764, 2002.

Angermann, L. Balanced apostriori error estimates for finite volume type discretizations of convection-dominated ellpitic problems. *Computing*, 55: 305–323, 1995.

Apel, Th.; Lube, G. Anisotropic mesh refinement for a singularly perturbed reaction diffusion model problem. *Appl. Numer. Math.*, 26: 415–433, 1998.

Astrid, P. Reduction of process simulation models: a proper orthogonal decomposition approach. *PhD thesis, Technische Universiteit Eindhoven*, Eindhoven, 2004.

Bakhvalov, N.S. Towards optimization of methods for solving boundary value problems in the presence of boundary layers. *Zh. Vychisl. Mat. Mat. Fiz.*, 9: 841–859, 1969.

Bangerth, W.; Rannacher, R. Adaptive Finite Element Methods for Differential Equations. *Birkhauser Verlag*, 2003.

Baronas, R.; Šimkus, R. Modelling the bacterial self-organization in circular container along the contact line as detected by bioluminescence imaging, *Nonlinear Anal. Model. Control*, 16(3): 270–282, 2011.

Beckett, G.; Mackenzie, J. On a uniformly accurate finite-difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution. *J. Comput. Appl. Math.*, 131: 381–405, 2001.

Berkooz, G.; Holmes, P.; Lumley, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1), 539–575, 1993.

Boglaev, I.P. The numerical solution of a nonlinear boundary value problem with small parameter effecting the highest derivative. *Zh. Vychisl. Mat. Mat. Fiz.*, 24: 1649–1656, 1984.

Brenner, M.P.; Levitov, L.S.; Budrene, E.O. Physical mechanisms for chemotactic pattern formaton by bacteria. *Biophys. J.*, 74(4): 1677–1693, 1998.

Budrene, E.O.; Berg, H.C. Dynamics of formaton of symmetrical patterns by chemotactic bacteria. *Nature*, 376(6535): 49–53, 1995.

Cai, L.; White, R. E. Reduction of model order based on proper orthogonal decomposition for lithium-ion battery simulations. Journal of The Electrochemical Society, 156(3): A154–A161, 2009.

Cakir, M.;Amiraliyev, G. A finite difference method for the singularly perturbed problem with nonlocal boundary condition. *Appl. Math. Comput.*, 160: 539–549, 2005.

Cazabat, A.; Heslot, F.; Troian, S.; Carles, P. Fingering instability of thin spreading films driven by temperature gradients. *Nature*, 346: 824–826, 1990.

Chevalier, C.; Pellegrini, F. PT-Scotch: A Tool for Efficient Parallel Graph Ordering. *Parallel Computing*, 34(6-8): 318–331, 2008.

Čiegis, R. Numerical solution of a problem with small parameter for the highest derivative and a nonlocal condition. *Lith. Math. Journal*, 28(1): 90–96, 1988.

Čiegis, R.; Ilgevičius, A.; Liess, H.; Meilūnas, M.; Suboč, O. Numerical simulation of the heat conduction in electrical cables. *Mathematical modelling and analysis*, 12(4): 425–439, 2007.

Čiegis, Raim.; Čiegis, Rem.; Meilūnas, M.; Jankevičiūtė, G.; Starikovičius V. Parallel numerical algorithm for optimization of electrical cables. *Mathematical modelling and analysis*, 13(4): 471–482, 2008.

Čiegis, R.; Bugajev, A. Numerical approximation of one model of the bacterial self-organization. *Nonl. Anal. Model. Contr.*, 17(3): 253–270, 2012.

Čiegis, R.; Iliev, O.; Starikovičius, V.; Steiner, K. Numerical algorithms for solving problems of multiphase flows in porous media. *Math. Model. Anal.*, 11(2): 133–148, 2006.

Dongping, Zhang. Optimierung zwangsgekühlter Energiekabel durch dreidimensionale FEM-Simulationen. *Dissertation, Universität Duisburg-Essen*, 2009.

Eberl, H.J.; Sudarsan, R. OpenACC Parallelisation for Diffusion Problems, Applied to Temperature Distribution on a Honeycomb Around the Bee Brood: A Worked Example Using BiCGSTAB. *Parallel Processing and Applied Mathematics Lecture Notes in Computer Science*, 311–321, 2014

Eisenbach, M. Chemotaxis. *Imperial College Press*, London, 2004.

Electricity of France. Code_Saturne. [seen: Aug. 2015]. URL: http://code-saturne.org.

Engl, H.W.; Hanke, M.; Neubauer, A. *Regularization of Inverse Problems*. Kluwer, Dordrecht, 1996.

Eriksson, K.; Estep, D.; Hansbo, P.; Johnson, C. Introduction to adaptive methods for differential equations. *Acta Numerica*, 5: 105–158, 1995.

Evans, M.; Žurič, Z.; Basara, B.; Frolov S. A novel SIMPLE-based pressure-enthalpy coupling scheme for engine flow problems. *Math. Model. Anal.*, 17(1): 1–20, 2012.

Falk, R.; Osborn, J. Remarks on mixed finite element methods for problems with rough coefficients. *Math. Comp.*, 62: 1–19, 1994.

Fic, A.; Białecki, R. A.; Kassab, A. J. Solving transient nonlinear heat conduction problems by proper orthogonal decomposition and the finite-element method. *Numerical Heat Transfer, Part B: Fundamentals*, 48(2): 103–124.

Gerisch, A.; Chaplan, M.A. Robust numerical methods for taxis-diffusion-reaction systems: Applications to biomedical problems. *Math. Comput. Model.*, 43: 49–75, 2006.

Gerisch, A.; Verwer, J. Operator splitting and approximate matrix factorization for taxis-diffusion-reaction models. *Appl. Numer. Math. Comput. Model.*, 42: 159–176, 2002.

GPGPU. General-Purpose computation on Graphics Processing Units. [seen: Aug. 2015]. URL: http://gpgpu.org

Herlihy, M.; Luchangco, V. Distributed computing and the multicore revolution, *ACM SIGACT News*, 39(1): 62–72, 2008

Higuera, P., Lara, J., Losada, I. Realistic wave generation and active wave absorption for Navier-Stokes models: Application to OpenFOAM. *Coastal Engineering*, 71(1): 102–118, 2013.

Hoshino, T. ; Maruyama, N. ; Matsuoka, S. ; Takaki, R. CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 136–143, 2013.

Hundsdorfer, W. Numerical Solution of Advection-Diffusion-Reaction Equations *Lecture notes for Ph.D. course*, Thomas Stieltjes Institute, 2000.

Ilgevicius, A. Analytical and numerical analysis and simulation of heat transfer in electrical conductors and fuses. *Dissertation, Universität der Bundeswehr München*, 2004.

Ilgevicius, A.; Liess, H.D. Calculation of the heat transfer in cylindrical wires and electrical fuses by implicit finite volume method. *Mathematical Modelling and Analysis*, 8: 217–228, 2003.

Il'in, V.P. High order accurate finite volumes discretization for Poisson equation. *Siberian Math. J.*, 37(1): 151–169, 1996.

Incropera, F.; DeWitt, P.; David, P. Introduction to heat transfer. *John Willey & Sons*, New Yourk, 1985.

Hairer, E.; Norset, S.P.; Wanner, G. *Solving Ordinary Differential Equations I – Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer, Berlin, 1993.

Hairer, E.; Wanner, G. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, 1996.

Hillen, T.; Painter, K.J. A users guide to PDE models for chemotaxis. *J. Math. Biol.*, 58(1-2): 183–217, 2009.

Hundsdorfer, W.; Verwer, J.G. *Numerical Solution of Time-Dependent Advection-Difusion Reaction Equations*, volume 33 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, New York, Tokyo, 2003.

John, V. A numerical study of a posteriori error estimators for convection-diffusion equations. *Comput. Methods Appl. Mech. Eng.*, 190: 757–781, 2000.

Kadalbajoo, M.; Gupta, V. A brief survey on numerical methods for solving singularly perturbed problems. *Appl. Math. Comput.*, 10(3): 3641–3716, 2010.

Kancleris, Ž.; Šlekas, G.;Čiegis, R. Sensitivity of asymmetrically necked planar structure for microwave pulse power measurement in rectangular waveguide *IEEE Sensors Journal*, 13(4): 1143–1147, 2013.

Karahan, M.; Kalenderli, O. Coupled Electrical and Thermal Analysis of Power Cables Using Finite Element Method. *Heat Transfer – Engineering Applications*, 2011.

Karimi, K.; Dickson, N. G.; Hamze, F. A performance comparison of CUDA and

OpenCL. *arXiv preprint arXiv:1005.2581*, 2010.

Karypis, G. METIS, [seen: Aug. 2015]. URL: http://glaros.dtc.umn.edu/gkhome/views/metis.

Keller, E.; Segel, I. Initiation of slime mold aggregation viewed as instability. *J. Theor. Biol.*, 26: 399–415, 1970.

Kumar, V.; Grama, A.; Gupta, A.; Karypis, G. *Introduction to parallel computing: design and analysis of algorithms*, Benjamin/Cummings, Redwood City, 1994.

Kunert, G. A posteriori h^1 error estimation for a singularly perturbed reaction-diffusion problem on anisotropic meshes. *IMA J. Numer. Anal.*, 25(2): 408–428, 2005.

LeVeque, R; Erratum, Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 32: 1704, 1995.

LeVeque, R. Clawpack. [seen: Aug. 2015]. URL: http://www.clawpack.org.

LeVeque, R. *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, 2002.

Eymard, R.; Gallouet, T.; Herbin, R. The finite volume method, *update of the preprint n0 97-19 du LATP, UMR 6632, Marseille, September 1997 which appeared in Handbook of Numerical Analysis, P.G. Ciarlet, J.L. Lions eds, vol 7, pp 713–1020*, 2003.

Linß, T. Maximum-norm error analysis of a non-monotone fem for a singularly perturbed reaction-diffusion problem. *BIT Numer. Math.*, 47: 379–391, 2007.

Liuge, D.; Kang, L; Fanmin, K. Parallel 3D finite difference time domain simulations on graphics processors with Cuda. *Computational Intelligence and Software Engineering*, 1–4, 2009.

Loudyi, D; Falconer, R.A.; Lin, B. Mathematical development and verification of a non-orthogonal finite volume model for groundwater flow applications. *Adv. Water Res.*, 30: 29–42, 2007

Lyushnin, A.; Golovin, A.; Pismen, L. Fingering instability of thin evaporating liquid films. *Physical Rev. E.*, 65(2): 021602, 2002.

Marquez, A., Oviedo, J. J., Odloak, D. Model reduction using proper orthogonal decomposition and predictive control of distributed reactor system. *Journal of Control Science and Engineering*, 3, 2013.

Marowka, A. Parallel computing on any desktop *Communications of the ACM*, 50(9): 74–78, 2007

Muhammad, N.; Eberl, H.J. OpenMP Parallelization of a Mickens Time-Integration

Scheme for a Mixed-Culture Biofilm Model and Its Performance on Multi-core and Multi-processor Computers *High Performance Computing Systems and Applications Lecture Notes in Computer Science*, 5976: 180–195, 2010

Neher, J. H.; McGrath, M. H. The Calculation of the temperature rise and load capability of cable systems,. *AIEE Transactions*, Part III, Volume 76, 752–772, 1957.

Popinet, S. Gerris. [seen: Aug. 2015]. URL: http://gfs.sourceforge.net.

Mackenzie, J. Uniform convergence analysis of an upwind finite-difference approximation of a convection-diffusion boundary value problem on an adaptive grids. *IMA J. Numer. Anal.*, 19(2): 233–249, 1999.

Micikevicius, P. 3D finite difference computation on GPUs using CUDA. *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, 79–84, 2009.

Nho Hao, D.;Van Duc, N.; Lesnic, D. Regularization of parabolic equations backward in time by a non-local boundary value problem method. *IMA J. Appl. Math.*, 75: 291–315, 2010.

Murray, J.D. *Mathematical Biology I: An Introduction*. Springer, Berlin, 2002.

Murray, J.D. *Mathematical Biology II: Spatial Models and Biomedical Applications*. Springer, Berlin, 2003.

Makhkamova, I. Numerical Investigations of the Thermal State of Overhead Lines and Underground Cables in Distribution Networks. *Doctoral thesis, Durham University*, 2011.

OpenFOAM Foundation. OpenFOAM. [seen: Aug. 2015]. URL: http://www. openfoam.com.

OpenFVM. [seen: Aug. 2015]. URL: http://openfvm.sourceforge.net.

Pankratius, V.; Tichy, W. Software engineering for multicore systems: an experience report *IWMSE '08 Proceedings of the 1st international workshop on Multicore software engineering*, 53–60, 2008

Petit, O.; Bosioc, A.; Nilsson, H.; Muniean S.; Susan-Resigo, R. Unsteady simulations of the flow in a swirl generator using OpenFOAM. *International Journal of Fluid Machinery and Systems*, 4(1): 199–208, 2011

Piscaglia, F.; Montorfano, A.; Onorati, A. Development of fully-automatic parallel algorithms for mesh handling in the OpenFOAM-2.2.x technology, *SAE Technical Paper 2013-24-0027*, (2013),

Qiu, Y.; Sloan, D. Analysis of difference approximations to a singularly perturbed two-point boundary value problem on an adaptively generated grid. *J. Comput. Appl. Math.*, 101: 1–25, 1999.

Rivera, O.; Furlinger, K.; Kranzimuller, D. Investigating the scalability of OpenFOAM for the solution of transport equations and large eddy simulations, *Lecture Notes in Computer Science*, 7017: 121–130, 2011

Roos, H.; Linß, T. Sufficient conditions for uniform convergence on layer-adapted grids. *Computing*, 63: 27–45, 1999.

Samarskii, A.A. *The Theory of Difference Schemes*. Marcel Dekker, Inc., New York–Basel, 2001.

Shishkin, G. Grid Approximation of Singularly Perturbed Elliptic and Parabolic Equations. *PhD thesis, Keldysh Institute*, Moscow, 1990.

Šimkus, R.; Kirejev, V.; Meškienė, R.; Meškys, R. Torus generated by Escherichia coli. *Exp. Fluids*, 46(2): 365–369, 2009.

Šlekas, G. Investigation of electrodynamic properties of small scale objects using finite-difference time-domain method. *PhD thesis, Center for Physical Sciences and Technology*, 2014.

Starikovičius, V.; Čiegis, R.; Iliev, O. A parallel solver for optimization of oil filters. *Math. Model. Anal.*, 16(2): 326–342, 2011.

Strang, G. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5: 506–517, 1968.

Strogatz, S.H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Perseus Books Publishing, New York, 1994.

Stynes, M.; Roos, H.; Tobiska, L. Numerical Methods for Singularly Perturbed Differential Equations. *Springer*, Berlin, 1996.

Palacios, F.; Economon, T. SU2. [seen: Aug. 2015]. URL: http://su2.stanford.edu.

Sun, P.; Luo, Z.; Zhou, Y. Some reduced finite difference schemes based on a proper orthogonal decomposition technique for parabolic equations. *Applied Numerical Mathematics*, 60(1): 154–164, 2010.

Taler, J.; Duda, P. Solving Direct and Inverse Heat Conduction Problems. *Springer*, Berlin, 2006.

Thomaszewski, B.; Pabst, S.; Blochinger, W. Exploiting parallelism in physically-based simulations on multi-core processor architectures *EGPGV '07 Proceedings of the 7th Eurographics conference on Parallel Graphics and Visualization*, 69–76, 2007

Tichonov, A.N.; Samarskii, A.A. Homogeneous finite difference schemes. *Zh. Vychisl. Mat. Mat. Fiziki*, 1(1): 5–63, 1961.

Tikhonov, A.N.; Arsenin, V.Y. *Solutions of Ill Posed Problems*. V.H. Winston and Sons, New York, 1977.

Touvet, T.; Balmforth, N.; Craster, R.; Sutherland, B. Fingering instability in buoyancy-driven fluid filled cracks. *J. Fluid Mech.*, 672: 60–77, 2011.

Tumanova, N.; Čiegis, R. Predictor-corrector domain decomposition algorithm for parabolic problems on graphs. *Math. Model. Anal.*, 17(1): 113–127, 2012.

University of Florida. aCute. [seen: Aug. 2015]. URL: http://www.cise.ufl.edu/ ungor/aCute.

Vulanovic, R. A priori meshes for singularly perturbed quasilinear two-point bounday value problems. *IMA J. Numer. Anal.*, 21(1): 349–366, 2001.

Vulanovic, R. Mesh Construction for Discretization of Singularly Perturbed Boundary Value Problems. *PhD thesis, University of Novi Sad*, Novi Sad, 1986.

Vulanovic, R. The layer-resolving transformation and mesh generation for quasilinear singular perturbation problems. *J. Comput. Appl. Math.*, 203: 177–189, 2007.

Wahl, P.; Ly Gagnon, D. S.; Debaes, C.; Van Erps, J.; Vermeulen, N.; Miller, D. A. B.; Thienpont, H. B-calm: an open-source multi-gpu-based 3D-FDTD with multi-pole dispersion for plasmonics. *Progress In Electromagnetics Research*, Volume 138, 467–478, 2013.

Weller, H. G.; Tabor, G.; Jasak, H.; Fureby, C. A Tensorial Approach to Computational Continuum Mechanics Using Object-oriented Techniques. *Journal of Computational Physics*, 12(6): 620–631, 1998.

Wang, H. H. A parallel method for tridiagonal equations, *ACM Trans. Math. Software*, 7(2): 170–183, 1981.

List of Scientific Publications by the Author on the Topic of the Dissertation

Articles in Peer-reviewed Journals

Bugajev, A; Čiegis, R. 2012. Comparison of adaptive meshes for a singularly perturbed reaction-diffusion problem, *Mathematical modelling and analysis: the Baltic journal on mathematical applications, numerical analysis and differential equations*, 17(5):732–748, ISSN 1392-6292 [ISI Web of Science].

Čiegis, R; Bugajev, A. 2012. Numerical approximation of one model of bacterial self-organization, *Nonlinear analysis: modelling and control*, 17(3): 253–270, ISSN 1392-5113 [ISI Web of Science].

Jankevičiūtė, G; Leonavičienė, T; Čiegis, R; Bugajev, A. 2013. Reduced order models based on pod method for Schrödinger equations, *Mathematical modelling and analysis: the Baltic journal on mathematical applications, numerical analysis and differential equations*, 18(5):694–707, ISSN 1392-6292 [ISI Web of Science].

Čiegis, R; Suboč, O; Bugajev, A. 2014. Parallel algorithms for three-dimensional

parabolic and pseudoparabolic problems with different boundary conditions, *Nonlinear analysis: modelling and control*, 19(3): 282–395, ISSN 1392-5113 [ISI Web of Science].

Bugajev, A; Jankevičiūtė, G; Suboč, O; Tumanova, N. 2014. The mathematical modelling of heat transfer in electrical cables, *Electrical, control and communication engineering*, 5: 46–53, ISSN 2255-9140.

Articles in Other Editions

Čiegis, R; Bugajev, A. 2013. Parallel numerical algorithms for simulation of multidimensional ill-posed nonlinear diffusion-advection-reaction models, in *Applied Parallel and Scientific Computing: proceedings of 11th International Conference, PARA 2012, Helsinki, Finland, June 13–15, 2012. Lecture Notes in Computer Science.* Berlin: Springer, 7782: 387–397, ISSN 0302-9743.

Čiegis, R; Bugajev, A; Kancleris, Ž; Šlekas, G. 2014. Parallel numerical algorithms for simulation of rectangular waveguides by using GPU, in *Parallel processing and applied mathematics: 10th international conference, PPAM 2013, Warsaw, Poland, September 8–11, 2013. Revised Selected Papers, Part II. Lecture Notes in Computer Science.* Berlin: Springer, 8385: 301–310, ISSN 0302-9743.

Čiegis, R; Starikovičius, V; Bugajev, A. 2014. On parallelization of the OpenFOAM-Based solver for the heat transfer in electrical power cables, in *Euro-Par 2014: Parallel Processing Workshops Euro-Par 2014 International Workshops, Porto, Portugal, August 25–26, 2014. Revised Selected Papers, Part I. Lecture Notes in Computer Science.* Switzerland: Springer, 8805: 1–11 ISSN 0302–9743.

Čiegis, R; Starikovičius, V; Bugajev, A. 2014a. On efficiency of the OpenFOAM-based parallel solver for the heat transfer in electrical power cables, in *Proceedings of the first international workshop on Sustainable Ultrascale Computing Systems (NESUS 2014), August 27–28, Porto, Portugal, 2014.* Madrid: Computer Architecture, Communications, and Systems Group (ARCOS), 43–46.

Summary in Lithuanian

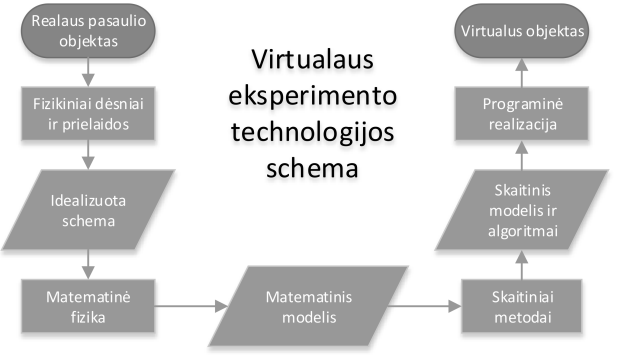
Įvadas

Motyvacija

Informacinėms technologijoms sparčiai besivystant jų galimybių išnaudojimas tampa vis didesniu iššūkiu. Kompiuterių greitaveika auga, atsiranda nauji algoritmai, programinės įrangos bibliotekos sudėtingėja vis sudėtingesnės. Vienas svarbiausių šių technologijų taikymų yra fizikinių procesų modeliavimas virtualaus eksperimento būdu. Virtualaus eksperimento technologijos schema pavaizduota 1S pav. Disertacijoje yra nagrinėjamos trys schemos (1S pav.) dalys:

- Skaitiniai metodai.
- Skaitiniai modeliai ir algoritmai.
- Programinė realizacija.

Disertacija yra skirta virtualaus eksperimento technologijai vystyti ir pagerinti. Darbe yra nagrinėjamas modeliavimo metodų taikymas programinės įrangos kūrimo kontekste. Pagrindinis šiame darbe nagrinėjamas mokslinis, abstrakčiai suformuluotas (vėliau jis bus patikslintas) klausimas yra: kaip pritaikyti kai kurias (vėliau tai bus patikslinta) šiuolaikines naujas technologijas kuriant programinę įrangą, kurios pagalba norima spręsti konkrečius naujus modeliavimo uždavinius. Darbe nagrinėjami šie klausimai:



1S pav. Virtualaus eksperimento technologijos schema

- Metodo taikymo efektyvumas. Metodas turi tenkinti skaitinių paklaidų reikalavimus ir jo realizacijos veikimas turi būti pakankamai greitas.
- Kaip gerai yra išnaudojami kompiuterio ištekliai (teorinių pajėgumų konvertavimas į realius skaičiavimus).
- Gautų rezultatų stabilumas – jie turi būti teisingi skirtingoms nagrinėjamo uždavinio variacijoms (pvz., medžiagų savybių koeficientai ir sričių geometrija skirtingais atvejais gali smarkiai skirtis).

Pažymėtina, kad nėra universalių atsakymų į visus šiuos klausimus – atsakymai priklauso nuo konkretaus uždavinio ypatumų. Skirtingi fizikiniai procesai gali pasižymėti skirtingais unikaliais ypatumais, todėl yra svarbūs skirtingi taikomo metodo aspektai. Rinkoje yra daug plataus taikymo spektro modeliavimo paketų, kurie remiasi Baigtinių elementų metodu (BEM), Baigtinių tūrių metodu (BTM), Baigtinių skirtumų metodu (BSM). Tačiau universalieji paketai negarantuoja efektyvaus sprendimo visais atvejais. Rezultatų patikimumas taip pat priklauso nuo vartotojo – jis gali rinktis kai kuriuos nustatymus, kurių dažnai užtenka tam tikrų standartinių uždavinių sprendimui. Tačiau sudėtingiems uždaviniams būdingi ypatumai, dėl kurių uždavinio sprendimas darosi sudėtingas, neefektyvus arba neįmanomas. Kompiuteriams greitėjant atsiranda galimybė ir poreikis spręsti vis daugiau sudėtingų uždavinių, todėl yra daug galimybių modeliavimo paketuose įdiegtoms technologijoms gerinti. Šitokio pobūdžio problemoms spręsti ir yra skirta ši disertacija. Fizikinių proceso modeliavimo realizacija kuriama atliekant kelis žingsnius, kiekvienas iš jų turi būti atliktas korektiškai, kad užtikrinti modeliavimo rezultatų ir realių reiškinių kokybišką ir kiekybišką atitikimą. Šie žingsniai yra:

1. Realaus pasaulio reiškinys aprašomas idealizuota schema. Klaidingos prielaidos duos neteisingus rezultatus.

2. Idealizuota schema aproksimuojama diferencialinėmis ir/arba integralinėmis lygtimis. Diferencialinės lygtys turi korektiškai aprašyti idealizuotos schemos proceso komponentus ir jų tarpusavio sąryšius. Kai kurios papildomos prielaidos ir supaprastinimai gali kokybiškai paveikti modeliavimo procesą ir iškreipti rezultatus.

3. Specialieji skaitiniai metodai ir algoritmai pritaikomi gauto matematinio modelio sprendimui. šie algoritmai privalo būti efektyvūs ir stabilūs. Yra svarbu parinkti algoritmus atsižvelgiant į modelį arba jo parametrus.

4. Skaitiniai algoritmai realizuojami programinėje įrangoje, gaunamas virtualus objektas. Realizacija gali būti neefektyvi ir kartais ją reikia adaptuoti prie tam tikrų kompiuterinės architektūros ypatumų. Šioje disertacijoje tai susiejama su paskirstytųjų skaičiavimų sistemomis. Tokių sistemų galimybėms išnaudoti yra būtina sukurti algoritmo modifikaciją – lygiagrečiąją algoritmo versiją.

Šiame darbe nagrinėjami 3 ir 4 punktai sprendžiant tris modeliavimo problemas. Pirmajame uždavinyje matematinio modelio lygtys yra sprendžiamos naudojant trijų dimensijų Baigtinių skirtumų laiko srities metodą (3D BSLS), ypatingas dėmesys skirtas efektyviems skaičiavimams naudojant GPU (ang. Graphics Processing Units, lietuviškai tai yra vaizdo apdorojimo įrenginiai) įrenginius. Antroji problema yra apie uždavinio sprendimą su specifine sričių geometrija, dideliais koeficientų trūkiais. Uždavinį galima praplėsti iki daugiafizikinio proceso modeliavimo, arba jis gali būti kito optimizavimo uždavinio dalis, dėl to labai išauga skaičiavimų apimtis. Mes siūlome efektyvų tokio uždavinio sprendimo kelią, tame tarpe ir naudojant paskirstytuosius skaičiavimus. Trečioji problema kyla modeliuojant bakterijų saviorganizacijos procesus, kai formuojasi ypatingi raštai ir neįmanoma valdyti algoritmo stabilumo ir įvertinti paklaidas naudojant klasikinę matematinio modeliavimo teoriją. Mes siūlome patikimus algoritmus ir jų lygiagrečiąsias versijas, formuluojame išvadas tolimesniems technologijų tobulinimams šia kryptimi. Pažymėtina, kad nagrinėjant visas tris problemas panaudojami lygiagretieji skaičiavimai, kurie užima labai svarbią vietą disertacijos rezultatuose.

Problemos formulavimas

Darbe yra naudojami standartiški skaitiniai metodai: Baigtinių elementų metodas (BEM), Baigtinių tūrių metodas (BTM) ir Baigtinių skirtumų metodų (BSM). Tačiau šių metodų taikymas tam tikrais atvejais yra neefektyvus ir būtina kurti technologijas, kurios optimizuotų programinės įrangos veikimą. Tyrimas apima šiuos tris atvejus:

1. GPU technologijos taikymas trijų dimensijų baigtinių skirtumų laiko srities metodo (3D BSLS) realizacijai yra nepakankamai ištirtas: nėra bendro skaičiavimo laiko modelio, kuris tiktų skirtingiems GPU įrenginiams ir skirtingiems metodo variacijoms.

2. Uždavinių su sudėtingomis geometrijos sritimis ir dideliais lygčių koeficientų trūkiais sprendimas naudojant Baigtinių tūrių metodo (BTM) paketus tam tikrais atvejais yra neefektyvus tame tarpe ir naudojant paskirstytuosius skaičiavimus.
3. Modeliuojant procesą, aprašantį sudėtingų struktūrų formavimą, matomi efektai tam tikrais atvejais kokybiškai priklauso ne tik nuo modelio, bet ir metodų. Tačiau yra būtina užtikrinti matomų efektų kokybinę priklausomybę tik nuo modelio, parenkant stabilius skaitinius metodus.

Darbo aktualumas

Kompiuterinių technologijų potencialui augant virtualaus eksperimento technologijos naudojimas, kai fizikinis eksperimentas pakeičiamas skaitine simuliacija, darosi vis populiaresnis. Geresnės technologijos suteikia naujas galimybes – sudėtingesniems modeliams tirti, geresniam rezultatų tikslumui pasiekti. Skaičiavimų pajėgumai yra vienas pagrindinių ribojančių faktorių virtualių eksperimentų realizacijai. Todėl yra labai svarbu užtikrinti efektyvų kompiuterinių išteklių panaudojimą. Vis sudėtingesnių procesų nagrinėjimas kelia algoritmų korektiškumo ir patikimumo klausimų. Efektyviam kompiuterinių išteklių išnaudojimui algoritmus yra būtina modifikuoti ir adaptuoti skaičiavimams specialiuose skaičiavimų įrenginiuose.

Tyrimų objektas

Tyrimų objektą sudaro fizikinių reiškinių modeliavimo paskirstytieji algoritmai realizuojantys baigtinių tūrių ir baigtinių skirtumų metodus.

Darbo tikslas

Disertacijos tikslas – tam tikrus fizikinių reiškinių modeliavimo uždavinius (jų grupę) padaryti efektyviai ir patikimai išsprendžiamus, naudojant paskirstytųjų skaičiavimų algoritmus.

Darbo uždaviniai

Darbo uždaviniai išskaidyti į tris grupes:

- Uždaviniai skirti GPU įrenginių panaudojimui, realizuojant 3D BSLS metodą:
  1. Atlikti 3D BSLS skaičiavimų su GPU realizacijos galimybių analizę.
  2. Pasiūlyti efektyvų algoritmą tokiems uždaviniams spręsti naudojant GPU įrenginius CUDA platformoje.

3. Atlikti 3D BSLS skaičiavimų su GPU detaliąją analizę. Sukurti skaičiavimų laiko prognozės modelį.
4. Pritaikyti sukurtą technologiją taikomojo uždavinio sprendimui.

- Uždaviniai skirti efektyviam BTM taikymui, kai modeliavimo uždavinys apibrėžtas sudėtingoje sričių geometrijoje ir medžiagų koeficientai yra trūkūs:
  1. Atlikti BTM realizacijos galimybių analizę.
  2. Atlikti skaičiavimus su pasirinktu BTM paketu, pritaikius jį šilumos laidumo uždaviniams vidutinės ir didelės galios elektriniuose kabeliuose. Ištirti rezultatų kokybę, identifikuoti sprendimo trūkumus ir apribojimus, susijusius su modeliuojamos problemos ypatumais.
  3. Pagerinti modeliavimo kokybę modifikuojant paketą ir/arba kuriant papildomas priemones.
  4. Ištirti gautų sprendimų patikimumą ir efektyvumą sprendžiant uždavinį paskirstytųjų skaičiavimų priemonėmis.
- Nestabilaus proceso modeliavimas, kai formuojasi sudėtingos sprendinio struktūros.
  1. Atlikti bakterijų saviorganizacijos modeliavimą (Eisenbach (2004)), kai procesas pasižymi specialių struktūrų formavimų, naudojant stabilius metodus.
  2. Ištirti algoritmo išlygiagretinimo galimybes tokio tipo uždaviniams, pasiūlyti sprendimą.
  3. Suformuluoti išvadas ir rekomendacijas tokių procesų modelius realizuojančiai programinei įrangai kurti.

Tyrimų metodika

Nagrinėjant darbo objektą buvo taikyti šie metodai: skaitinės schemos iš matematinio modeliavimo teorijos, konvergavimo analizė iš algoritmų teorijos, išplečiamumo analizė iš lygiagrečiųjų skaičiavimų teorijos, regresinė analizė iš statistikos teorijos.

Darbo mokslinis naujumas

GPU jau buvo plačiai taikomas 3D BSLS problemoms spręsti, kai kurie paskutiniai tyrimai šioje srityje (Wahl, Ly Gagnon, Debaes, Van Erps, Vermeulen, Miller, Thienpont (2013)) rodo, kad tokio tipo skaičiavimai gali būti atliekami naudojant kelis GPU įrenginius specialios programinės įrangos pagalba. Šioje disertacijoje aprašytiems tyrimams atlikti skaičiavimų realizacija yra kuriama naudojant CUDA platformą. Pastebėtina, kad yra galimybė realizuoti skaičiavimus naudojant OpenCL standartą, tačiau

kitų autorių tyrimai parodė, kad naudojant NVIDIA vaizdo plokštes OpenCL nėra efektyvesnis už CUDA (Karimi, Dickson, Hamze (2010)).

Skaičiavimus atliekantis algoritmas ir jo realizacija yra sukurta kitų autorių jau atliktų tyrimų pagrindu (Micikevicius (2009)). Šios disertacijos tyrimų naujumas yra tame, kad mes atlikome detaliąją skaičiavimų išplečiamumo analizę ir sukūrėme modelį skaičiavimų laikams prognozuoti. Taip pat mes atlikome testus su skirtingų kartų GPU įrenginiais. Taip pat parodome, kaip gauti empiriškai suformuluotus algoritmus remiantis bendrais lygiagrečiųjų skaičiavimų teorijos principais, tai algoritmą leidžia tirti naudojant klasikinę lygiagrečiųjų algoritmų teoriją. Taip pat mes pritaikome realizaciją specifinei skaičiavimų schemai sukurtai praktinei problemai spręsti (Šlekas (2014)).

Antroji tyrimų dalis yra skirta šilumos laidumo elektriniuose kabeliuose modeliavimui. Šis uždavinys yra labai aktualus ir nagrinėtas kitų autorių (pvz., Karahan, Kalenderli (2011); Makhkamova (2011)), tačiau šiame darbe pagrindinis dėmesys telkiamas į algoritmų efektyvumą, leidžiantį kurti programinę įrangą efektyviai išnaudojančią kompiuterinius išteklius. Matematinis modeliavimas atliekamas naudojant BTM metodą. Konvergavimo analizė parodė, kad pasiūlytas sprendimo kelias duoda antrąją konvergavimo eilę $p = 2$, kas yra BTM teorinis maksimumas, nepaisant to, kad dėl uždavinio specifikos buvo naudojamas neortogonalus diskretizavimo tinklas.

šiems rezultatas pasiekti mes naudojame srities diskretizaciją specialiais trikampiais(ang. acute), kai baigtinių tūrių centrus galima imti Voronojaus taškuose. Taip pat tinklo singuliarumams valdyti buvo sukurtas naujas originalus tinklo adaptavimo algoritmas. Sprendiklio lygiagrečioji versija yra kuriama automatiškai pasitelkiant OpenFOAM paketo priemones, tačiau šis tyrimas parodė, kurie parametrai (iš didelio OpenFOAM parametrų sąrašo) ir kaip jie turi būti naudojami efektyviems skaičiavimams atlikti. Tyrimas atliktas šilumos laidumo uždavinio sprendimui skirtos programinės įrangos kūrimo kontekste, tačiau rezultatai gali būti taikomi plačiam panašių uždavinių spektrui.

Trečioje tyrimų dalyje mes parodome, kad bakterijų saviorganizacijos modeliavimo metu pasireškiantis nestabilumas yra siejamas su parabolinių uždavinių su atvirkštine laiko tėkme nekorektiškumu. Modelis aproksimuotas stabiliais diskrečiais metodais. Skaitinių eksperimentų rezultatai parodo, kad tokiems uždaviniams problemoms negalima taikyti klasikinio skaitinių algoritmų konvergavimo kriterijų. Vietoje maksimumo ir panašių klasikinių paklaidų metrikų tokiais atvejais reikia naudoti kokybinius kriterijus (panašiai, kaip atraktorių teorijoje) arba vidurkintus statistinius rodiklius (kaip taikant diskrečiųjų dalelių metodą). Pasiūlyta algoritmų lygiagrečioji versija, pateikti išplečiamumo analizės rezultatai.

Darbo rezultatų praktinė reikšmė

3D BSLS metodas yra plačiai taikomas sprendžiant Maksvelo lygtis. Kuriant programinę įrangą ir norint padaryti ją konkurencingą rinkoje yra svarbu išnaudoti naujas technologines galimybes, tokias kaip skaičiavimus su GPU. Kai skaičiavimuose yra naudojami GPU įrenginiai svarbu ištirti skirtingų GPU įtaką skaičiavimų greičiui, nes

rinkoje yra didelis spektras įvairių įrenginių, kurių pasirinkimas dažniausiai yra atliekamas neatsižvelgiant į konkrečią programinę įrangą. Didžiąją rinkos dalį sudaro bendros paskirties vaizdo plokštės su bendrojo taikymo grafikos apdorojimo įrenginiais (ang. general-purpose graphics processing units (GPGPU)). Yra nemažai tyrimų atliktų naudojant profesionalius brangius įrenginius orientuotus į skaičiavimus (Tesla GPU). Tačiau, kai informatikos inžinieriams reikia orientuotis į vidutinius rinkos vartotojus, tampa svarbu kurti programinę įrangą atsižvelgiant į vaizdo plokštes su bendrojo taikymo grafikos apdorojimo vienetais (GPGPU). Taip pat yra ypatingai svarbu tirti įvairių GPGPU įrenginių įtaką rezultatams. Būtent šią kryptį ir nagrinėjame disertacijoje, rezultatai gali būti panaudoti programinės įrangos inžinierių praktiniais tikslais. Taip pat mes formalizuojame kai kurias svarbias GPU skaičiavimo charakteristikas darydami jas labiau prieinamomis tiems, kas orientuojasi į apibendrintus, matematinėmis formulėmis išreikštus receptus. atiduoda pirmenybę matematinėms formuluotėms. Disertacijoje parodome, kad kitų autorių suformuluotas algoritmas skirtas 3D BSLS skaičiavimams su GPU (Micikevicius (2009)) gali būti išvedamas iš bendrų paskirstytųjų skaičiavimų principų.

Efektyvus šilumos laidumo modeliavimas didelės galios elektros kabeliuose yra svarbus sprendžiant industrinius optimizavimo uždavinius. Darbe pasiūlyti technologiniai sprendimai gali būti naudojami kuriant programinę įrangą, kurios paskirtis yra kabelių dizaino ir elektros pernešimo tinklų infrastruktūros optimizavimas, jau turimos infrastruktūros optimizavimas. Optimizavimo uždaviniams yra kritiška spręsti tiesioginį uždavinį labai efektyviai, nes optimizavimo proceso metu sprendimo procedūra gali būti kviečiama daug kartų. Kai kurios problemos gali pareikalauti didžiulės skaičiavimų apimties, galinčios pareikalauti savaičių tiems skaičiavimams atlikti net naudojant superkompiuterius. Šiuo metu elektros linijų ir kabelių projektavimas yra pripažintos svarbiomis kryptimis optimizavimui atlikti. Disertacijoje gauti rezultatai taip pat nesudėtingai gali būti pritaikyti plataus spektro kitokiems uždaviniams spręsti, tai nesiriboja vien šilumoje laidumo kabeliuose modeliavimu. Lygiagrečiųjų sprendiklių versijų testai parodė, kad pasiūlyti sprendikliai yra efektyvūs daugiabranduolinėse ir daugiakompiuterinėse sistemose.

Efektyvaus ir patikimo technologinio sprendimo kūrimas yra būtinas, kai modeliuojamas fizikinis procesas yra nestabilus klasikinio stabilumo kriterijaus (iš matematinio modeliavimo teorijos) prasme. Šie kriterijai buvo naudojami tiriant standartiškus skaitinius diferencialinių lygčių sprendimo metodus ir jų konvergavimas yra įrodytas naudojant klasikines paklaidų normas – būtent todėl prieš programinės įrangos kūrimą yra būtina ištirti šį nestabilaus modelio atvejį gyliau. Disertacijoje yra ištirta kaip simuliuoti situaciją, kai sprendinys formuoja stochastinius raštus. Toks tyrimas yra būtina sąlyga kokybiškai programinei įrangai kurti tokio tipo procesų simuliacijai, nes metodai turi užtikrinti, kad nestabilumas ir sprendimo jautrumas būtų sąlygotas modelio, o ne metodų. Taip pat mes pasiūlėme lygiagrečiąją algoritmo versiją, tam atvejui jeigu iš programinės įrangos bus reikalaujama pagreitinti skaičiavimus naudojant lygiagrečiuosius algoritmus.

Ginamieji teiginiai

1. Išnagrinėtas algoritmas skirtas šabloninio tipo skaičiavimų realizacijai GPU įrenginiuose, pasiūlytas skaičiavimų laiko prognozės modelis nustato 3D BSLS (Šlekas 2014) skaičiavimų laiko įvertį, kuris yra tos pačios eilės kaip ir tikrų skaičiavimų, t.y. nesiskiria daugiau nei 10 kartų.
2. Pasiūlyti algoritmai specifinių šilumos laidumo uždavinių sprendimui yra efektyvūs, nes išlaiko BTM teorinį konvergavimo greitį. Taip pat jie efektyviai išnaudoja kompiuterių išteklius, kai naudojamas kompiuterių klasteris.
3. Specifiškas nestabilus procesas formuoja sudėtingas struktūras dėl modelio nestabilumo, todėl yra svarbu stabiliųjų skaitinių algoritmų panaudojimas, kas užtikrina metodų kokybinių efektų įtakos rezultatams nebuvimą. Pasiūlyti algoritmai yra matematiškai patikimi, tačiau specialios metrikos (normos) yra būtinos norint įvertinti gautų rezultatų tikslumą.

Darbo rezultatų aprobavimas

Rezultatai pristatyti 7 tarptautinėse 3 nacionalinėse ir 3 jaunųjų mokslininkų konferencijose. Disertacijos tema paskelbti 9 moksliniai straipsniai. 5 iš jų yra publikuoti periodiniuose recenzuojamuose mokslo žurnaluose, iš jų 4 yra įtraukti į Thompson Reuters ISI Web of Science duomenų bazę ir turi citavimo indeksą, 1 – įtrauktas į kitas duomenų bazes. 3 straipsniai yra klasifikuojami kaip ISI Conference Proceedings. Detalusis sąrašas pateiktas disertacijos skyriuje „List of Scientific Publications by the Author on the Topic of the Dissertation“.

Disertacijos struktūra

Disertaciją sudaro įvadas, 3 skyriai, išvados, literatūros sąrašas, autoriaus publikacijų disertacijos tema sąrašas, santrauka lietuvių kalba. Skyriai padalinti į poskyrius, poskyriai – į skyrelius. Naudojama numeracija „skyrius.poskyris.skyrelis“, „skyrius.paveikslas“, „skyrius.lentelė“. Formulių numeracija yra atskira kiekvienam skyriui, „skyrius.formulė“.

Disertacijoje yra 129 puslapiai be priedų, 201 formulė, 53 paveikslai ir 20 lentelių. Disertacijoje cituojami 106 informacijos šaltiniai.

Padėka

Esu dėkingas mano darbo vadovui prof. habil. dr. Raimondui Čiegiui už vadovavimą mano doktorantūros studijų procesui. Jo vadovavimas doktorantūros studijų metu sudarė itin palankias sąlygas mano mokslinės kompetencijos augimui, kas yra pagrindinis šių studijų tikslas.

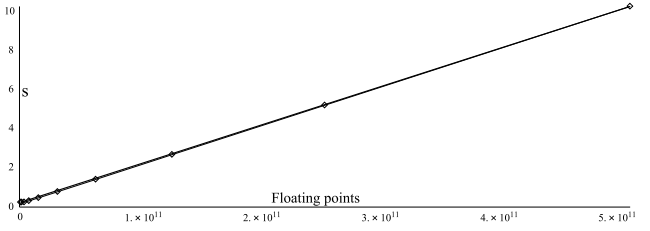
Taip pat dėkoju kolegoms už pagalba atliekant tyrimus. Ypatingai norėčiau išskirti ir padėkoti dr. Gediminui Šlekui ir habil. dr. Žilvinui Kancleriui už bendradarbiavimą atliekant tyrimus ir prieigą prie skaičiuojamuoju išteklių.

Dėkoju recenzentams prof. dr. Olegui Vasilecui ir prof. dr. Zenonui Navickui už pagalbą darant vertingas disertacijos pataisas, doktorantūros skyriaus vedėjui doc. dr. Šarūnui Mikaliūnui už pagalbą tvarkant disertacijos tekstą.

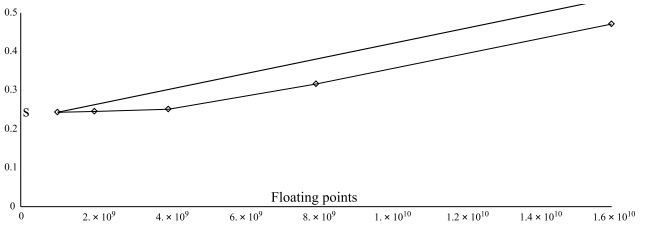
Darbo dalis buvo atlikta Eureka projekto E!6799 POWEROPT „Mathematical modelling and optimization of electrical power cables for an improvement of their design rules“ rėmuose.

1. 3D Baigtinių skirtumų laiko srities skaičiavimai naudojant grafinį procesorių

Pirmajame skyriuje parodoma, kaip efektyviai pritaikyti GPU technologiją trijų dimensijų Baigtinių skirtumų laiko srities metodo(3D BSLS) realizacijai. Realizuotas specialus algoritmas naudojant CUDA platformą ir pritaikytas 3D BSLS skaičiavimams. Pasiūlytas skaičiavimų laiko prognozės modelis, kuris teisingai nustato pagrindinę skaičiavimų laiko eilę (2S pav. ir 3S pav.). Parodytas realizacijos GPU išteklių efektyvus panaudojimas. Pasiektas geras skaičiavimų efektyvumas, gautas apie 10 kartų pagreitėjimas lyginant su CPU versija. Parodyta, kad skaičiavimų laiką ribojantis rodiklis paprastai yra GPU atminties pralaidumas. Naudojantis pasiūlyta realizacija buvo išspręstas realus technologinis uždavinys, rezultatai pristatyti kitoje disertacijoje (Šlekas (2014)).



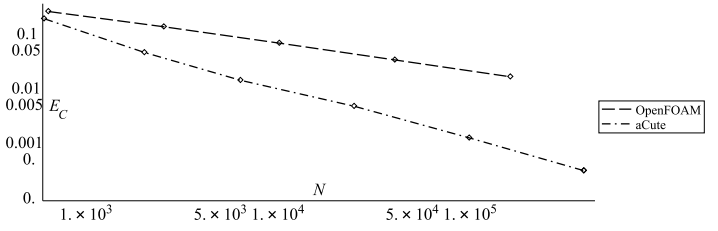
2S pav. Pasirinktos GPU vaizdo plokštės laikai esant skirtingiems skaičiavimu apimtimis (angl. floating points). Kreivė su taškais ant jos vaizduoja empirinius duomenis, storesnė tiesė vaizduoja pasiųlytą modelį



3S pav. Pasirinktos GPU vaizdo plokštės laikai esant skirtingiems skaičiavimu apimtimis (angl. floating points), priartintas didžiausio netikslumo vietoje vaizdas. Kreivė su taškais ant jos vaizduoja empirinius duomenis, storesnė tiesė vaizduoja pasiųlytą modelį

2. Šilumos laidumo kabeliuose modeliavimo tyrimas

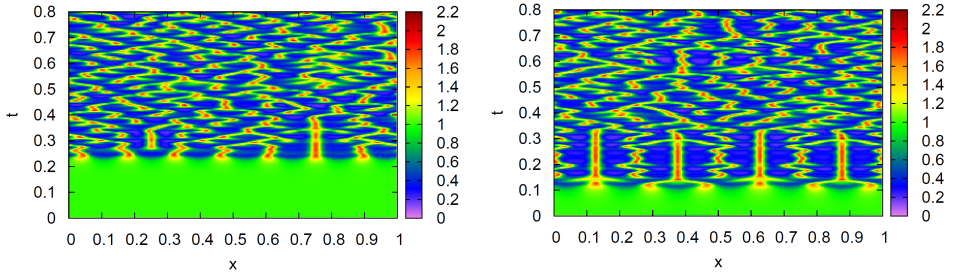
Antrajame skyriuje nagrinėjama, kaip efektyviai pritaikyti Baigtinių tūrių metodą (BTM) uždaviniams su sudėtingomis geometrijos sritimis ir dideliais lygčių koeficientų trūkiais ant sričių kontūrų. Skaičiavimams atlikti panaudota atviro kodo programinė įranga OpenFOAM. Rezultatai patikrinti naudojant nepriklausomą nuo OpenFOAM BTM sprendiklį rašytą C++ terpėje. Lokalizuoti OpenFOAM efektyvumo trūkumai, susieti su uždavinio geometrija ir medžiagų koeficientais. Disertacijoje išnagrinėtos dvi skirtingos srities diskretizacijos tinklų generavimo strategijos. Mūsų pasiūlyta tinklo generavimo strategija davė geresnius rezultatus už tuos kurie gauti naudojant į OpenFOAM integruotą strategiją (4S pav.). Pasiūlytai strategijai realizuoti OpenFOAM paketas buvo modifikuotas. Nustatyta, kad efektyviam suformuluoto uždavinio sprendimui adaptyvusis diskretizacijos tinklas yra būtinas, adaptyvusis laiko žingsnis yra rekomenduojamas. Pasiūlyta adaptyvaus tinklo generavimo euristika. Sukurtas ir ištirtas specialus algoritmas, apdorojantis tinklų singuliarumo atvejus. Taip pat uždavinys išspręstas naudojant lygiagrečiąją sprendiklio versiją, atlikti skaičiavimo eksperimentai VGTU klasteryje Vilkas. Atlikta detalioji išplečiamumo analizė.



4S pav. Palyginimas neortogonalių acute tinklų strategijos ir OpenFOAM BTM algoritmo su iteratyviomis korekcijomis neortogonaliems tinklams

3. Nestabilių procesų modeliavimo metodai

Trečiajame skyriuje nagrinėjamas modeliavimo stabilumas ir korektiškumas, kai modeliuojamas bakterijų populiacijos dinamikos procesas, aprašantis specialių raštų formavimą (Eisenbach (2004)). Atliktas bakterijų saviorganizacijos modeliavimas (5S pav.) ir parodytas nestabilumo ryšys su nekorektišku difuzijos modeliavimo uždaviniu su atvirkštine laiko tėkme. Matematinis modelis aproksimuotas skaitiniu modeliu naudojant patikimas stabilias skaitines schemas. Įrodyta, kad diskretusis sprendinys pagrindines savybės paveldi iš matematinio modelio. Parodyta, kad nestabilumas aprašo sprendinio vidinį elgesį, todėl tokiems procesams modeliuoti klasikiniai modeliavimo paklaidų vertinimo metodai netinka. Pasiūlyta algoritmo lygiagrečioji versija, pateikti išplečiamumo analizės rezultatai.



a) $u(x,0) = 1 + 0.2\sin(2\pi x)$, b) $u(x,0) = 1 + 0.2\sin(4\pi x)$

5S pav. Lastelių tankio $U(x,t)$ vizualizavimas esant skirtingoms pradinėms sąlygoms

Bendros išvados

1. Pasiūlytas skaičiavimų laiko prognozės modelis leidžia įvertinti skirtingus GPU įrenginius jų efektyvumo 3D BSLS skaičiavimų atžvilgiu. Tačiau modelį reikia modifikuoti norint jį pritaikyti naujesniems GPU įrenginiams.
2. Skaičiavimų laiką ribojantis rodiklis paprastai yra GPU atminties pralaidumas.
3. OpenFOAM turi efektyvumo trūkumų, susietų su uždavinio geometrija ir medžiagų koeficientais – iš keturkampių sudarytų tinklų ir neortogonalumo korekcijų algoritmų integruotų į OpenFOAM panaudojimas sumažina (maždaug nuo 2 iki 1) BTM konvergavimo eilę.
4. Disertacijoje pasiūlyta technologija išsprendžia efektyvumo problemas, susietas su srities geometrijos ypatumais – srities diskretizacija trikampiais, pasižyminčiais tam tikromis geromis savybėmis (visi kampai smailūs) ir baig-

tinių tūrių centrų skaičiavimas Voronojaus taškuose eliminuoja neortogonalumo paklaidas išlaikant optimalų BTM konvergavimo greitį.

5. Efektyviam suformuluoto uždavinio sprendimui adaptyvusis erdvinės srities diskretizacijos tinklas yra būtinas, adaptyvusis laiko žingsnis yra rekomenduojamas. Pasiūlyta adaptyvaus tinklo generavimo euristika išsprendžia adaptyvaus tinklo problemą.

6. Harmoninio vidurkio formulių naudojimas yra būtinas norint išlaikyti norimą tikslumą, dėl diferencialinės lygties koeficientų didelių trūkių ant posričių kontūrų. Šių formulių panaudojimas yra valdomas per integruotus į OpenFOAM valdomus parametrus.

7. Tinklų singuliarumo atvejus apdorojantis specialus algoritmas išsprendžia tinklo singuliarumo problemą.

8. Sukurto OpenFOAM pagrindu sprendiklio efektyvumas yra geras (efektyvumo koeficientas yra vieneto eilės) panaudojant visus 32 branduolius, kai yra nepaisoma duomenų išsaugojimo į failus kaštų (matuojamas laikas nevertinant kompiuterio darbo su failais) ir tinklo išskaidymui panaudotas mazgų apimties balansavimas.

9. Mažesnis klasterio mazgo lokalių uždavinių dydis pagreitina (nagrinėtu atveju apie 30%) lygiagrečiojo algoritmo skaičiuojamąją dalį dėl geresnio procesoriaus atminties panaudojimo.

10. Svarbu ištirti išsaugojimo į failus kaštus (matuojant kompiuterio darbo laiką su failais) tarpiniais laiko momentais, tai gali lemti skaičiavimų efektyvumo sumažėjimą, kai, pavyzdžiui, sprendinys yra išsaugojamas kas 5–10 laiko žingsnių.

11. Diskrečiojo sprendinio nestabilumas yra sąlygotas matematinio modelio.

12. Skaitinių eksperimentų rezultatai parodo, kad nestabilumas aprašo realų sprendinio elgesį ir tokiems nestabiliems procesams modeliuoti klasikiniai modeliavimo paklaidų vertinimo metodai netinka. Vietoje maksimumo ir panašių klasikinių paklaidų metrikų, tokiais atvejais reikia naudoti kokybinius kriterijus arba vidurkintus statistinius rodiklius.

13. Pasiūlyta algoritmo lygiagrečioji versija leidžia spręsti uždavinį naudojant paskirstytuosius skaičiavimus.

14. Vienmatis modelis gali būti pernelyg paprastas kokybiškam virtualaus eksperimento rezultatų ir realių eksperimentų atitikimui nustatyti. Todėl reikia ištirti modelio 2D ir 3D apibendrinimus.

Appendixes[1]

Appendix A. Adaptive Meshes
Appendix B. The Finite Difference Scheme
Appendix C. The Co-authors Agreements to Present Publications for the Dissertation Defence
Appendix D. The Copies of Scientific Publications by the Author on the topic of the Dissertation

[1]The annexes are supplied in the enclosed compact disc

Andrej Bugajev

THE INVESTIGATION OF EFFICIENCY
OF PHYSICAL PHENOMENA MODELLING
USING DIFFERENTIAL EQUATIONS
ON DISTRIBUTED SYSTEMS

Doctoral Dissertation
Technological Sciences,
Informatics engineering (07T)

FIZIKINIŲ REIŠKINIŲ MODELIAVIMO
NAUDOJANT DIFERENCIALINES LYGTIS
EFEKTYVUMO PASKIRSTYTOSE
SISTEMOSE TYRIMAS

Daktaro disertacija
Technologiniai mokslai,
informatikos inžinerija (07T)

2015 10 26. 11,5 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto leidykla „Technika“,
Saulėtekio al. 11, LT-10223 Vilnius, *http://leidykla.vgtu.lt*
Spausdino UAB „BMK leidykla“,
J. Jasinskio g. 16, 01112 Vilnius